НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

До захисту допущено

Завідувач кафедри
_____Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
"___"_____2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системне програмування»

спеціальності **123 «Комп'ютерна інженерія»**

на тему: Програмне забезпечення на основі операціональних трансформацій для створення користувацьких додатків

Виконав

студент IV курсу, групи КВ-61

Бровдій Євгеній Юрійович_____ _____
(прізвище, ім'я, по батькові) (підпис)

Керівник доц. каф. СПіСКС, к. т. н., доцент Тарасенко-Клятченко О.В._____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М._____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали)_____(підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ІАЛЦ.045440.001 ОА	Опис альбому	2	
3	А4	ІАЛЦ.045440.002 ТЗ	Технічне завдання	4	
4	А4	ІАЛЦ.045440.003 ТП	Відомість технічного проєкту	2	
5	А4	ІАЛЦ.045440.004 ПЗ	Пояснювальна записка	60	
6	А4	ІАЛЦ.045440.005 Д1	Схема структурна	1	
7	А4	ІАЛЦ.045440.006 Д2	Схема алгоритму	1	
8	А4	ІАЛЦ.045440.007 Д3	Схема структурна	1	
9	А4	ІАЛЦ.045440.008 Д4	Схема структурна	1	

				ДП ІАЛЦ 045440.000		
	ПІБ	Підп.	Дата			
Розробн.	Бровдій			Відомість дипломного проєкту	Лист	Листів
Керівн.	Тарасенко-Клятченко				1	1
Консульт.	Клятченко				КПІ ім. Ігоря Сікорського Каф. СПіСКС Гр. КВ-61	
Н/контр.	Клятченко					
Зав.каф.	Романкевич					

Пояснювальна записка
до дипломного проєкту

на тему: Програмне забезпечення на основі операціональних трансформацій для створення користувацьких додатків

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Системне програмування»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
«___»2020 р.

ЗАВДАННЯ
на дипломний проєкт студента
Бровдія Євгенія Юрійовича
(прізвище, ім'я, по батькові)

1. Тема проєкту «Програмне забезпечення на основі операціональних трансформацій для створення користувацьких додатків»,
керівник проєкту доц. каф. СПіСКС, к. т. н., доцент Тарасенко-КлятченкоО.В.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «__»_____ 2020 р. №_____

2. Термін подання студентом проєкту _____

3. Вихідні дані до проєкту:

- клієнтське програмне забезпечення на основі операціональних трансформацій у вигляді ОТ ядра для взаємодії з віддаленим серевером

- вебдодаток файлової системи на основі операціональних трансформацій.

4. Зміст пояснювальної записки

- аналіз існуючих розподілених систем управління даними;
- аналіз основних властивостей операціональних трансформацій;
- опис роботи модулів клієнтського програмного забезпечення на основі операціональних трансформацій;
- тестування взаємодії клієнтського програмного забезпечення з вебдодатком та аналіз результатів.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

- принцип взаємодії основних модулів клієнтського програмного забезпечення ОТ (схема структурна);
- операціональні перетворення клієнтської системи ОТ (схема алгоритму);
- принцип роботи програмного забезпечення для вебдодатків (схема структурна);
- принцип взаємодії модулів вебдодатку(схема структурна).

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М., к.т.н., доцент каф. СПіСКС		

7. Дата видачі завдання "30" жовтня 2019 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	18.11.2019	
2.	Розроблення та узгодження технічного завдання	30.11.2019	
3.	Аналіз існуючих рішень	10.01.2020	
4.	Підготовка матеріалів першого розділу проєкту	18.01.2020	
5.	Розроблення програмного забезпечення	16.02.2020	
6.	Відлагодження програмного продукту	14.03.2020	
7.	Підготовка матеріалів другого розділу проєкту	01.04.2020	
8.	Розроблення інтерфейсу програмного забезпечення	20.04.2020	
9.	Підготовка графічної частини	01.05.2020	
10.	Оформлення документації дипломного проєкту	14.05.2020	

Студент _____ Євгеній БРОВДІЙ
(підпис)

Керівник проєкту_____Оксана ТАРАСЕНКО-КЛЯТЧЕНКО
(підпис)

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (60 с., 24 рис., 1 табл., 4 додатки).

Об’єкт розробки –клієнтське програмне забезпечення на основі технології операціональних трансформацій, яке дозволяє швидко, зручно та безпечно обмінюватись даними будь-якого типу між клієнтом та сервером операціональних трансформацій, передаючи серверу дані та операції у бінарному форматі для подальшого їх зберігання у вигляді графа з комітами.

В роботі розроблено програмне забезпечення, а саме вебдодаток для роботи з файловою системою, який дозволяє зберігати та видаляти файли та їх дані. Користувач має змогу працювати зі своєю власною файловою системою з кількох пристроїв одночасно, навіть при тривалій відсутності підключення до мережі.

Розроблене клієнтське програмне забезпечення з використанням операціональних трансформацій дозволяє:

- надсилати клієнтські запити на віддалений сервер операціональних трансформацій;
- перетворювати дані будь-якого типу в бінарний формат та формувати зліпки даних для серверної частини;
- вирішувати клієнтські конфлікти даних;
- застосовувати основні властивості системи операціональних трансформацій для перетворення кількох операцій ;

В ході виконання дипломного проєкту:

- розроблено вебдодаток файлової системи на основі ОТ технології;
- проведено аналіз існуючих рішень;
- розроблено структуру клієнтського ядра операціональних трансформацій.

Ключові слова: ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, РОЗПОДІЛЕНІ СИСТЕМИ, ОПЕРАЦІОНАЛЬНІ ПЕРЕТВОРЕННЯ, ГРАФ ОПЕРАЦІОНАЛЬНИХ ТРАНСФОРМАЦІЙ, JAVASCRIPT, JSON, ВЕБ-ДОДАТОК, МОДЕЛЬ ДАНИХ, ОТ ТЕХНОЛОГІЯ, ВІДДАЛЕНИЙ СЕРВЕР.

ABSTRACT

Qualification work includes an explanatory note (60 pages, 24 pictures, 1 table, 4 appendices).

The object of development is client software based on operational transformation technology, which allows fast, convenient and secure exchange of data of any type between the client and the operational transformation server, transmitting data and operations to the server in binary format for further storage as a graph of commits.

The software has been developed, namely a web application for working with the file system, which allows you to save and delete files and their data. The user can work with his own file system from several devices at the same time, even if there is no network connection for a long time.

Developed client software using operational transformations allows:

- send client requests to a remote server of operational transformations;
- convert data of any type into binary format and form data casts for the server part;
- resolve customer data conflicts;
- apply the basic properties of the system of operational transformations to transform several operations;

During the implementation of the diploma project:

- developed a web application of a file system based on OT technology;
- the analysis of existing decisions is carried out;
- the structure of the client core of operational transformations is developed.

Keywords: SOFTWARE, DISTRIBUTED SYSTEMS, OPERATIONAL TRANSFORMATIONS, GRAPH OF OPERATIONAL TRANSFORMATIONS, JAVASCRIPT, JSON, REMOTE SERVER.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045440.002 ТЗ	Програмне забезпечення	4		
			на основі операціональних			
			трансформацій для			
			створення користувацьких			
			додатків			
			Технічне завдання			
	А4	ІАЛЦ.045440.003 ТП	Програмне забезпечення	2		
			на основі операціональних			
			трансформацій для			
			створення користувацьких			
			додатків			
			Відомість технічного			
			проєкту			
	А4	ІАЛЦ.045440.004 ПЗ	Програмне забезпечення	60		
			на основі операціональних			
			трансформацій для			
			створення користувацьких			
			додатків			
			Пояснювальна записка			

Зм	Лист	№ докум.	Підп	Дата	ІАЛЦ.045440.001 ОА		
Розроб.		Бровдій Є.Ю.			Програмне забезпечення на основі операціональних трансформацій для створення користувацьких додатків *Опис альбому*	Літ.	Лист 1 / Листів 2
Перев.		Тарасенко -					
		Клятченко О.В.				НТУУ «КПІ ім. Ігоря Сікорського», ФПМ, КВ-61	
Н. контр.		Клятченко Я.М.					
Затв.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045440.005 Д1	Принцип взаємодії	1		
			основних модулів			
			клієнтського програмного			
			забезпечення ОТ.			
			Схема структурна			
	А4	ІАЛЦ.045440.006 Д2	Операціональні	1		
			перетворення клієнтської			
			системи ОТ.			
			Схема алгоритму			
	А4	ІАЛЦ.045440.007 Д3	Принцип роботи	1		
			програмного забезпечення			
			для вебдодатків.			
			Схема структурна			
	А4	ІАЛЦ.045440.008 Д4	Принцип взаємодії	1		
			модулів вебдодатку.			
			Схема структурна			
		Диск CD-ROM	Текст ПЗ. Тексти програм.	1		
			Графічний матеріал.			

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Програмне забезпечення на основі операціональних трансформацій для створення користувацьких додатків».

Галузь застосування: розробка користувацьких вебдодатків на основі сучасних інформаційних технологій.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проєкту є створення програмного забезпечення на основі операціональних трансформацій, яке будь-які вебдодатки могли б використовувати для взаємодії з віддаленим сервером.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

- Обробка запитів клієнтського програмного забезпечення на основі операціональних трансформацій в рамках клієнт-серверної архітектури;
- Застосування основних властивостей системи операціональних трансформацій;
- Сповіщення про виняткові ситуації;
- Масштабована архітектура;
- Відмовостійкість та здатність підтримувати велику кількість клієнтських запитів на сервер;

5.2. Вимоги до апаратного забезпечення

- Процесор: 4-ядерний процесор;
- Оперативна пам’ять: 16Гб;
- Наявність доступу до мережі Internet (Ethernet);

5.3. Вимоги до програмного та апаратного забезпечення користувача

- Будь-який пристрій зі здатністю відправляти http-запити;

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів
1.	Вивчення літератури за тематикою проєкту	15.04.2020
2.	Розроблення та узгодження технічного завдання	30.04.2020
3.	Аналіз існуючих рішень	05.05.2020
4.	Підготовка матеріалів першого розділу дипломного проєкту	10.05.2020
5.	Підготовка матеріалів другого розділу дипломного проєкту	18.05.2020
6.	Підготовка графічної частини дипломного проєкту	20.05.2020
7.	Оформлення документації дипломного проєкту	25.05.2020
8.	Попередній огляд матеріалів диплому на кафедрі	30.05.2020

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.045440.004 ПЗ	Програмне забезпечення	60		
			на основі операціональних			
			трансформацій для			
			створення користувацьких			
			додатків			
			Пояснювальна записка			
	А4	ІАЛЦ.045440.005 Д1	Принцип взаємодії	1		
			основних модулів			
			клієнтського програмного			
			забезпечення ОТ.			
			Схема структурна			
	А4	ІАЛЦ.045440.006 Д2	Операціональні	1		
			перетворення клієнтської			
			системи ОТ.			
			Схема алгоритму			
	А4	ІАЛЦ.045440.007 Д3	Принцип роботи	1		
			програмного забезпечення			
			для вебдодатків.			
			Схема структурна			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045440.003 ТП		
Розробив		Бровдій Є.Ю.			Програмне забезпечення на основі операціональних трансформацій для створення користувацьких додатків Відомість технічного проекту	Літ.	Аркуш 1 / Аркушів 2
Перевірила		Тарасенко - Клятченко О.В.				НТУУ «КПІ ім. Ігоря Сікорського», ФПМ, КВ-61	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045440.008 Д4	Принцип взаємодії	1		
			модулів вебдодатку.			
			Схема структурна			
		Диск CD-ROM	Текст ПЗ. Тексти програм.	1		
			Графічний матеріал.			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045440.003 ТП	Арк. 2

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ПЗ – програмне забезпечення.

BLOB (BinaryLargeObject) – бінарний набір даних, що розглядається як єдиний, неперервний об’єкт.

CRDT (Conflict-FreeReplicatedDataType) – типи даних, які можна скопіювати на багато мережевих вузлів і оновлювати паралельно без координації між вузлами.

Git – програмне забезпечення для контролю версій, яке використовується для роботи з файлами.

Global-OT – технологія роботи з даними, що базується на основі операціональних трансформацій і була розроблена на мові Java з використанням фреймворку Datakernel.

JSON (JavaScriptObjectNotation) – текстовий формат обміну даних, оснований на JavaScript, зручний для читання, запису для людини та комп’ютера.

IPFS (InterPlanetaryFileSystem) – протокол та система розповсюдження даних за змістом, яка побудована на концепції ідентифікування даних за хешом, тобто за ID, який розраховується й відповідно залежить, від їхнього внутрішнього значення.

LSM-дерево – це структура даних, що використовується в багатьох базах даних та забезпечує швидкий доступ до даних по індексу в умовах частого надходження запитів на вставку.

OLAP (OnlineAnalyticalProcessing) – технологія обробки даних, що полягає в підготовці сумарної результуючої інформації на основі великих структуризованих масивів даних.

OLTP (OnLineTransactionProcessing) - онлайнова обробка транзакцій; спосіб організації баз даних, при якому система працює з невеликими за

розмірами транзакціями, що йдуть великим потоком, і при цьому клієнту потрібний від системи максимально швидкий час відповіді.

OT (OperationalTransformation) / ОП (Операціональні перетворення) – технологія, що використовується для автоматичного рішення конфліктів у складних системах за допомогою перетворення операцій над даними.

OT-System – система управління даними на основі операціональних трансформацій.

OT-Node – програмне забезпечення для роботи з системою операціональних трансформацій, яке дозволяє робити серверні запити та працювати з будь-якими типами даних.

P2P (Peer-to-Peer) – однорангова децентралізована мережа, основана на рівноправності учасників.

P2Pe (PointtoPointEncryption) - протокол шифрування даних, що використовується поверх з'єднань PPP; стандарт, установлений Радою по стандартам безпеки PCI (PaymentCardIndustryDataSecurity Standard) для підвищеної безпеки проведення електронних транзакцій.

React.js – фреймвок на мові Javascript для написання веборієнтованих додатків.

XML (Extensible Markup Language) -

запропонований консорціумом WorldWideWebConsortium (W3C) стандарт побудови мов розмітки ієрархічно структурованих даних для обміну між різними застосунками, зокрема, через Інтернет.

ВСТУП

Проблеми злагодженої роботи розподіленої системи існують на всіх рівнях програмного забезпечення (ПЗ) та обладнання. Окремовебсистеми додають труднощі з мережами. Тому існує безліч підходів для подолання цих проблем і немає жодного універсального.

Зазвичай, вебдодатки мають клієнт-серверну архітектуру, яка дозволяє легко передавати дані з клієнту (додатку) на сервер, зберігати їх там та отримувати в зручному форматі. Передача даних відбувається в єдиній мережі, в якій знаходяться клієнт та сервер. Якщо стан мережі нестабільний, або ж виникають ситуації, що вводять клієнта в офлайн режим, то це не повинно вплинути на цілісність та правильність передачі даних. Коли мережа знову стабілізується, можуть виникнути конфлікти між даними, з якими працювали клієнти. Існує чимало способів уникнення таких конфліктів при зберіганні даних на серверах. Одним з них є використання розподілених систем.

Наразі існує кілька видів розподілених систем для роботи з даними у таких додатках. Такі системи характеризуються децентралізацією обробки даних та дозволяють багатьом клієнтам легко працювати з будь-якими даними одночасно, навіть в режимі офлайн. Клієнти матимуть можливість паралельно оновлювати дані, з якими вони працюють, без координації між мережевими вузлами.

Такий підхід нині є передовим в області інформаційних технологій, адже він дозволяє забезпечити надійну синхронізацію даних між користувацькими додатками в мережі Інтернет.

1. АНАЛІЗ ІСНУЮЧИХ РОЗПОДІЛЕНИХ СИСТЕМ УПРАВЛІННЯ ДАНИМИ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЄКТУ

1.1. Причини використання розподілених систем для управління даними

Розподілена система - це набір незалежних комп'ютерів, що представляється їх користувачам єдиною об'єднаною системою. Такій системі притаманні певні особливості. Всі комп'ютери є автономними та незалежними один від одного. Це означає що вони можуть самостійно приймати та оброблювати інформацію незалежно від стану інших комп'ютерів та способу зберерігання інформації на них. В розподілених системах від користувачів приховані відмінності між комп'ютерами і способи зв'язку між ними. Про зовнішню організацію розподілених систем можна сказати аналогічним чином. Для користувачів всі комп'ютери є просто однаковими машинами для прийому, зберігання та передачі даних. В таких системах важливим є спосіб, за допомогою якого користувачі і додатки можуть однаково працювати в розподілених системах, окремо один від одного [1]. Їм не потрібно знати коли, де і як відбувається їх взаємодія. Такий вид систем здатний легко піддаватися масштабуваню та розширенню. Це вказує на факт існування ланки незалежних один від одного комп'ютерів, проте приховує реалізацію того, як ці комп'ютери насправді об'єднуються в єдину неподільну систему (рис. 1.1).

Рисунок 1.1 – Взаємодія незалежних комп’ютерів в розподіленій системі

Побудувати таку систему стало можливим завдяки розвитку мережних технологій та факту винаходу високошвидкісних комп’ютерних мереж. Спочатку були розроблені локальні мережі (LAN), які дозволили з’єднувати сотні комп’ютерів та обмінюватися інформацією між ними за кілька мікросекунд. Передача значних об’ємів даних стала загальнодоступною по всьому світу після об’єднання цих локальних мереж у сформовані глобальні мережі (WAN), які вже дозволяли обмін даними між мільйонами, а не сотнями, комп’ютерів, які розташовані віддалено один від одного по всьому світу.

Як і в будь-якій системі, в розподіленій системі її компоненти можуть виходити з ладу або втрачати свою ефективність. Користувачі та самі додатки не мають бути інформовані про такі пошкодження в системі. В розподілених системах заміна компонентів системи, або ж додавання нових потрібних елементів ніяк не має вплинути на їх інкапсуляцію у використовуваних додатках [2].

До винайдення розподіленої системи усі додатки та налаштунки використовували централізовані або однопроцесорні системи, які представляли собою один єдиний комп’ютер та кілька віддалених терміналів

для забезпечення взаємодії інформації.Як і в будь-яких системах, в розподіленій системі існують певні переваги та недоліки, які було б доцільно розглянути. Саме значні переваги зумовили заміну централізованих систем розподіленими. Переваги є як в програмній частині, так і в апаратній. Разом це і дозволило вирішити проблему швидкодії, абстрактності та доступності під час роботи користувачів з додатками у такій мережевій системі.

Переваги розподілених систем у програмній частині в порівнянні з централізованими системами організації взаємодії даних:

1) Легка масштабованість.

Це означає, що в разі розширення налаштунків на основі розподілених систем нові вимоги до інформаційної системи буде легко реалізувати.

2) Інтегрування наявних рішень.

Не потрібно розроблювати додаткову функціональність системи для того, щоб нові приєднані системні компоненти змогли користуватися системою.

3) Оптимізоване системне розширення.

Данний функціонал мінімізує ризик перевантаження окремих компонентів розподіленої системи.

4) Організаційне керування потужністю.

Забезпечує надійну, безперешкодну адаптованість та гнучкість самої системи. При чому реалізація даної переваги є ефективною з погляду вартості.

5) Довільна реконфігурація системи її власником [3].

Власник системи має доступ до всіх її компонентів, може вільно реконфігурувати систему так, як потрібно йому або іншим користувачам, для підвищення оптимізаційних характеристик (швидкодії тощо).

6) Автономність компонентів системи.

У разі виникнення помилки або пошкодження роботи одного з компонентів системи, це ніяк не вплине на її загальну роботу – так як усі

компоненти незалежні, то це не вплине на цілісність роботи системи. Система продовжить функціонувати так, як і раніше, розв’язуючи необхідні задачі.

7) Гнучкість використання даних.

Всі комп’ютери розподіленої системи можуть спільно використовувати дані та працювати з ними. Таким чином, створюється враження, що спільний ресурс під’єднаний до кожного комп’ютера окремо.

8) Доступність використання даних.

Дані можуть бути отримані користувачами швидко та незалежно від місця, де знаходяться самі користувачі. Причому розмір інформації, яку необхідно отримати, може бути будь-яким.

Також можна виокремити ряд переваг розподілених апаратних рішень, які були прийняті при створенні шаблону розподілених систем:

1) Простота модулів.

Дана перевага є досить комплексною, адже включає в себе кілька аспектів. Поєданння кількох модулів може забезпечити той самий результат і ту ж саму роботу з ресурсами, що і використання одного великого модуля. Проте, єдиний великий модуль буде дорожчим, складнішим в реалізації та буде менше піддаватися тестуванню, ніж сукупність окремих надійних модулів [4].

2) Загальна надійність.

При виході з ладу одного модуля система лише трохи втратить свою загальну продуктивність, проте на її функціональність це ніяк не вплине – інші робочі модулі просто заберуть на себе функції неробочого модуля.

3) Легке підвищення продуктивності.

В такій системі її продуктивність не буде залежати від кількості її апаратних елементів, а тим більше від розробки нових.

4) Оптимізоване тепловідведення.

Через особливість розміщення блоків на опорній платі будемо спостерігати полегшене тепловідведення в порівнянні з іншими системами.

5) Оптимізований температурний режим.

Реалізація зменшених абсолютних значень температур та зменшення диференціації температурних зон всередині всієї системи.

Через досконалу організацію ефективної взаємодії окремих частин розподіленої системи маємо і ряд недоліків. Розробляти такі системи стало набагато складніше, адже зросла сама обчислювальна складність системи та змінився принцип роботи з інформацією. З'явилась необхідність створення окремих модулів синхронізації, адже всі компоненти системи уже є незалежними і немає центрального керування всім процесом обробки даних. При виході з ладу якогось блоку чи окремого модуля – треба відслідковувати саме цей модуль та ізольовувати його [5]. Раніше це було непотрібним, оскільки модуль був один – великий і складний, який, умовно, міг бути лише в робочому та неробочому режимах. Також, при великій кількості модулів, з'явилась додаткова необхідність продумувати розподіл потужностей в такій системі, адже загальна потужність значно зросла. Через це потрібно вводити додаткові компоненти, що призвело до збільшення розмірів опорних плат.

1.2. Аналіз існуючих розподілених систем управління даними

Розподілені системи спрощують інтеграцію різних прикладних програм з різних комп'ютерів і є гарно масштабованими при правильному проектуванні. Існує багато типів розподілених систем. Усі вони створені щоб виконувати різні задачі, кожна система здатна вирішувати певний ряд інформаційних проблем та проблем роботи з даними в мережах. Можна виділити 3 основні класи розподілених систем управління даними. У кожного класу є своя специфіка роботи з даними в мережі та своє певна архітектура самої розподіленої системи. Для кращого розуміння дамо коротку характеристику кожного з цих класів.

До першого віднесемо розподілені обчислювальні системи, які в свою чергу можна поділити на кластерні, решіткові та хмарні. Такі обчислювальні системи являють собою набір подібних процесорів або робочих станцій, які запускають однакові операційні системи, з'єднані високошвидкісною локальною мережею LAN.

До другого класу можна віднести інформаційні розподілені системи. Вони можуть бути різними та використовуватись для різних цілей. Найпопулярнішими є бізнес-орієнтовані системи, системи обробки транзакцій та системи загальної інтеграції додатків в мережі Internet. Їх об'єднують саме принципи роботи з інформацією та можливість керування нею за допомогою чисельної кількості різноманітних операцій. Одночасне звернення до даних багатьма операціями в централізованих системах може призвести до конфліктів роботи з даними та їх пошкодження, особливо при відсутності доступу до мережі. Розподілені системи даного типу здатні вирішувати цю проблему та забезпечувати безконфліктні рішення при роботі з даними від багатьох користувачів одночасно. Користувачі можуть використовувати різні версії своїх додатків, у них можуть бути різні версії операційних систем та баз даних. Усе це могло б привести до конфліктів управління даними при використанні неправильної системи їх обробки [6]. Дані про інші види розподілених систем буде наведено далі в розділі 2.2.

Третій клас розподілених систем являє собою так звані вбудовані поширювальні розподілені системи. Такі системи є досить абстрактними в описі, проте вони базуються на стабільності мережевих вузлів та з'єднання з мережею. Їх часто використовують як електронні системи моніторингу здоров'я пацієнтів в онлайн режимі, як сенсорні мережі на обчилювальних пристроях, або так звані «домашні» системи, які здатні без суттєвого втручання людини синхронізувати домашні прилади, системи контролю за світлом, мережею та багато іншого. Разом такі системи здатні утворювати розумне середовище обробки та взаємодії даних. Це стало можливим саме

завдяки розвику інформаційних технологій що базуються на використанні розподілених систем керування даними.

Щоб краще зрозуміти принципи їх роботи, а також проблеми з даними які вони вирішують наведемо кілька прикладів використання розподілених систем. Самий простим прикладом може бути мережа Internet, яка являє собою, фактично, простір для збереження документів за певними адресами. Користувач, знаючи адресу та ім’я документу, може безперешкодно знайти його в мережі. Усі документи зберігаються на окремих серверах, тому користувач знає, що вони знаходяться в різних місцях мережевого простору. Таким чином, спостерігаємо приклад розподіленості даних.

Як інший приклад, розглянемо мережу робочих станцій у відділі певної компанії. Користувачі можуть звертатися до файлів за постійним шляхом доступу. Усі файли будуть розміщені в одній централізованій системі, яка може приймати запити від багатьох комп’ютерів користувачів. Хоч така система має централізоване зберігання даних, проте вона є розподіленою через можливість доступу до інформації з декількох місць одночасно і без виникнення мережевих конфліктів. Причому, коли користувач вводить певну команду на отримання або запис даних, система обирає оптимальне місце для виконання потрібної дії. Тобто під час запиту на сервер може обиратись або найближча вільна робоча станція (сервер), або власний комп’ютер користувача. Цей приклад демонструє властивість доступності інформації та колаборативності.

Ще одним прикладом практичного застосування розподіленої системи керування даними може бути система автоматичної обробки онлайн-замовлень товарів в мережі Internet. Для користувача така система здається централізованою, адже він не бачить усіх аспектів реалізації місця обробки його запиту. Коли приходить запит на певний товар, він починає оброблюватися відразу кількома центрами роботи з даними (наприклад відділ доставки, відділ оплати, відділ бухгалтерії та інші). Усі ці центри взаємодіють

між собою, передаючи користувацькі дані один одному та оброблюючи інформацію так, як потрібно саме їм. Фактично, це і є ті самі модулі синхронізації в розподіленій системі, де кожен модуль здатен вирішувати свої задачі.

Говорячи про різні системи розподіленого типу управління даними, варто згадати Git - програмне забезпечення для контролю версій, яке використовується для роботи з файлами. В цьому програмному забезпеченні є вбудована система рішення конфліктів між даними, де дані – це просто звичайні файли. Проте, по своїй суті, Git не являє собою систему, а є просто інструментом для роботи з різними версіями файлів. Оскільки Git може працювати тільки з файлами – то це є його головним недоліком, оскільки сучасні системи обробки даних (наприклад, на основі операціональних трансформацій або на основі реплікації даних) можуть працювати з будь-якими типами даних і працюють з надійнішою системою рішення конфліктів даних.

1.3. Обгрунтування теми дипломного проєкту

У сучасному світі існує безліч різноманітних користувацьких додатків, завдяки якими можна вільно, швидко та в зручний спосіб обмінюватися інформацією між собою. З кожним роком таких додатків стає все більше, з’являється велика конкуренція, тому самі додатки мають бути конкурентоспроможними. Все, що необхідно користувачеві в додатку – це простий інтерфейс та швидкий обмін інформацією з іншими користувачами. Простий інтерфейс будь-якого додатку зробити не складно, адже даний аспект передбачає передачу інформації у зручний спосіб (наприклад, просто натиснувши одну або дві кнопки у вебдодатку). Більш складною частиною для реалізації являється саме швидкодія обміну даних, якість передачі даних

мережею та спосіб зберігання самих даних та їх доступність на серверній частині.

Говорячи про клієнт-серверну архітектуру додатків, яка зараз використовується у переважній більшості нових вебпроєктів, найпоширеніший спосіб зберігання даних - це зберігання їх у визначеній базі даних, до якої буде предоставлений доступ з одного або кількох серверів. В такому разі, користувач робить клієнтський запит на сервер, після чого сервер робить запит в базу даних і повертає з неї значення. Проте усі бази даних можуть мати певні недоліки, які значно впливатимуть на швидкодію або спосіб обробки даних та роботи з ними. Тому зараз при створенні будь-яких користувацьких додатків надзвичайно важливим є попереднє планування не просто архітектури, яка буде використовуватись, а й оптимізації роботи з даними у таких додатках.

Зі збільшенням кількості користувачів в певних вебдодатках зросла необхідність рішення конфліктів роботи з даними та конфліктів зберігання інформації на серверах. В даному дипломному проєкті знадобиться детальніше розуміння таких розподілених систем для роботи з даними, як OT-система (OT-System) на основі операціональнихтрасформацій та CRDT розподілена система з використанням реплікації даних. Для опису обрані саме ці системи, адже вони є найбільш схожими між собою в роботі з даними та в організації віддалених серверів, а також вирішують однакові конфлікти при роботі з даними. Метою даного проєкту є створення програмного забезпечення, яке буде використовуватись системою операціональних трансформацій в багатьох користувацьких додатках для автоматичного рішення конфліктів даних будь-якого типу.

Актуальність використання цих самих розподілених систем можна продемонструвати на прикладі. Припустимо, нехай євебдодаток, який являє собою звичайний онлайн-чат, де люди можуть вільно переписуватися, бачитися та обмінюватися будь-якою інформацією між собою. Зараз існує

безліч подібних додатків, і їх об’єднує саме стабільність роботи при наявності підключення до мережі. Проте, як тільки мережа стане недоступною, або будуть траплятися збої, обмін данними стане неможливим. Якщо користувач не помітить цього (а він і не повинен думати про цілісніть передачі даних мережею), то при обробці даних виникнуть конфлікти.

Можна розглянути інший, більш складний приклад. Нехай є багато-користувацький текстовий онлайн редактор документів, який всі одночасно мають можливість редагувати. Система, яка оброблює запити користувачів на редагування, повинна правильно реагувати на дії користувачів та не допускати ситуацій, в яких запити одних користувачів прийматься безконфліктно, а інші ігноруються. Найскладніший випадок – це редагування кількома користувачами одного документу без наявності підключення до інтернету. Раніше, в міру складності реалізації, розробити таку систему було неможливо, проте зараз є така можливість саме завдяки використанню систем операціональних трансформацій та розподілених систем реплікації даних.

Такі системи є відносно новими і з’явились лише кілька років тому. До деяких з них немає ґрунтовної документації, через що при створенні вебдодатків розробники вдаються до використання тільки тих архітектурних способів, які є популярними та поширеними в даній галузі. При цьому вони не вивчають усі переваги і недоліки даних технологій, а просто використовують те, що можна зробити швидко і забезпечити його працездатність.

1.4. Висновки

Розподілені системи є основною для розвитку багатьох сучасних інформаційних технологій, а також створення вебдодатків. Проаналізовано основні властивості розподілених систем, їх застосування, а також детально роглянуто їх переваги та недоліки, порівняно з іншими подібними системами.

2. АНАЛІЗ ТЕХНОЛОГІЙ РОЗРОБКИ КОРИСТУВАЦЬКИХ ДОДАТКІВ НА ОСНОВІ ОПЕРАЦІОНАЛЬНИХ ТРАНСФОРМАЦІЙ

2.1. Архітектура системи операціональних трансформацій

Операціональні трансформації (ОП) представляють собою технологію, яка використовується в передових системах управління даними і являє собою цілий комплекс функціональних можливостей для роботи з ними. З самого початку, дана технологія була розроблена лише для підтримки сумісності редагування різних простих текстових документів. Згодом дану технологію вдосконалили та почали використовувати для інших додаткових цілей. Можливості сумісного редагування документів значно зросли з появою в операціональних трансформацій здатності вирішувати будь-які користувацькі конфлікти, блокування операцій при роботі з даними та їх компресія, редагування XML та HTML документів, а також різних документів деревовидної структури. Завдяки розвитку вебтехнологій та створенню складних систем обробки даних на основі операціональних трансформацій, розробники користувацьких додатків почали використовувати розподілені системи з використанням ОТ у своїх проектах [7].

Багаторозробницьких компаній мали на меті почати використовувати технологію операціональних трансформацій та будувати свої користувацькі додатки на її основі, проте розробити такі додатки виявилось складніше, ніж вони передбачали, через базову складність самої архітектури таких додатків та відсутність загальної документації по розробці ПО на основі операціональних трансформацій. Її необхідно було продумувати завчасно та передбачати усунення всіх складнощів, які можуть виникнути при роботі з різними компонентами системи операціональних трансформацій. Така система мала б

дуже комплексну та складну архітектуру, адже необхідно було б якось синхронізовувати усі дії користувачів у мережі та використовувати великі потужності серверів для зберігання даних [8]. Архітектура системи операціональних трансформацій має забезпечувати швидкодію відповіді на всі запити між серверами і клієнтами, мати вбудовану комплексну підсистему для рішення усіх користувацьких конфліктів під час взаємодії даних додатків в мережі.

Система операціональних трансформацій передбачає, що кожний користувач локально створює якусь операцію роботи з даними, наприклад додавання чи віднімання того чи іншого числа або символу залежно від специфіки додатків. На сервері усі операції групуються та правильно поєднуються між собою, використовуючи спеціальні властивості перетворень. Важливим є те, що з віддалених сайтів на локальний сайт клієнту приходять вже готові дані з внесеними перетвореннями. Самі ж перетворення гарантують, що відправлені дані не будуть спотворені, збережуть свою цілісність та логіку застосування, а також збережеться узгодженість даних між усіма користувачами в їх додатках. Властивість неблокування дає можливість вебдодаткам бути нечутливими до мережевих затримок: локальний час відгуку на запити є мінімальним та не залежить від мережі. Урахування усіх цих особливостей операціональних трансформацій важливе для побудови цілісної системи, з якою будуть працювати усі користувачі додатків.

На основі операціональних трансформацій можна розробити додатки для роботи з будь-якими типами даних, адже сама технологія передбачає написання розробником алгоритмів рішення конфліктів між потрібними типами. Найпопулярнішого застосування операціональні трансформації набули саме у додатках сумісного редагування текстових онлайн-документів, проте їх також використовують для при створенні простих лічильників та будь-яких додатків з сумісними до бази даних операціями. Проектування простих лічильників на основі операціональних трансформацій передбачає

наявність базових операцій роботи з лічильником (наприклад додавання та віднімання числового значення). Для їх створення використовують OLAP бази даних з семантикою комітів даних (подібно до системи Git) (рис. 2.1).

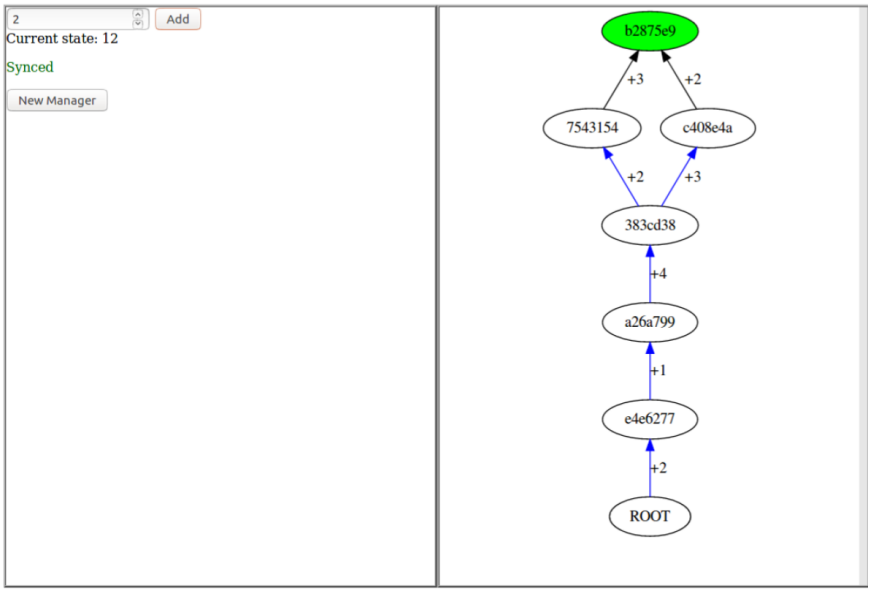

Рисунок 2.1 – Граф операціональних трансформацій для простого лічильника

Говорячи про побудову різноманітних додатків з сумісними до бази даних операціями, мається на увазі використання в самих додатках CRUD (Create-Read-Update-Delete) операцій, що приміняються до даних конкретного типу [9]. Для створення таких додатків за один з можливих способів реалізації беруть за основу OLTP бази даних з семантикою комітів. Керуючись простою логікою, видно, що в будь-якому додатку користувач повинен мати могу щось створювати, редагувати, видаляти або просто читати якісь дані. З точки зору проектування вебдодатків, саме ці операції є найдоречнішими при застосуванні операціональних трансформацій, адже все що може робити користувач за даними буде реалізовано.

Операціональні трансформації по своїй суті просто являють собою правильне рішення конфліктів даних згідно алгоритму, визначеного користувачем. Розлянемо кілька основних випадків виникнення конфліктів

даних та способи їх рішення. Для наглядності виникнення конфліктів візьмемо вебдодаток для спілкування та обміну текстовою інформацією між користувачами – звичайний мережевий чат. Два користувачі додатків просто спілкуються між собою в чаті, після чого в одного (або в обох відразу-складніший випадок) пропадає Інтернет. Не помітивши цього, користувачі продовжили спілкуватись і надіслали один одному вже кілька повідомлень одночасно. Після цього виникає ситуація мережевого конфлікту даних, адже операції обох користувачів мають примінитись. Результуюча версія чату, в якому знаходяться користувачі, повинна зберегти однотипність та узгодженість даних при появі мережі. Версія чату, на якій розійшлись користувачі,реплікована для них обох. Якщо один користувач зробив операцію видалення даних, а інший зробив додавання та видалення, при неправильному алгоритмі злиття або його відсутності на сервері буде мережевий конфлікт, в результаті якого обидва користувачі отримають неправильні дані, а також їх версія чату буде відрізнятись, що є недопустимим. Щоб не допустити конфліктів, операції кожного з користувачів накопичуються та відправляються на їх особисті репозиторії [10]. Репозиторії зв’язані із загальною системою серверів, на яких відбувається коректне злиття даних. Кожен сервер працює зі своїм набором даних, причому на серверах усі дані зберігаються у вигляді складних графів. Завдяки тому, що сервер знає пріоритетність застосування користувацьких операцій до набору даних, не виникає мережевих конфліктів (рис. 2.2). Після появи мережі обидва користувачі отримують однакову версію текстового чату або ж документу.

Рисунок 2.2 – Розв’язання конфліктів даних в додатку за допомогою ОТ

Іншим прикладом виникнення конфліктів даних в додатках може бути простий додаток для створення та редагування своїх власних нотаток. В такому додатку користувач сам взаємодіє з даними і ніхто крім нього не буде їх редагувати, проте складність конфліктів даних, які можуть виникнути, залишається такою самою, як при значній кількості користувачів. Уявімо ситуацію, в якій створюються, видаляються або редагуються нотатки одночасно з кількох пристроїв. При раптовій відсутності мережі виникнуть конфлікти злиття даних, адже на пристроях будуть різні версії нотаток в офлайн режимі. На одному пристрої нотатки можуть бути відредаговані, а на іншому взагалі видалені зі списку [11]. В цьому випадку знову потрібно використовувати пріоритетність виконання операцій при роботі з конкретними даними. Якщо операція видалення буде мати більший пріоритет ніж операція редагування, то нотатку буде видалено в результуючій версії після злиття (рис. 2.3).

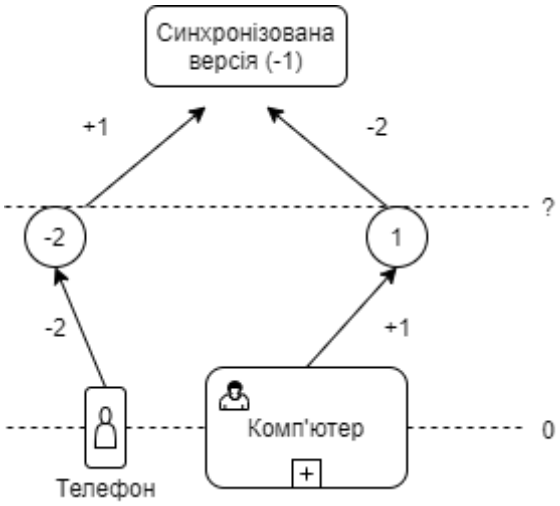

Рисунок 2.3 – Конфлікт роботи з даними на кількох пристроях одночасно

Важливим є те, що сервер звикористанням операціональних трансформацій завжди намагається не відкидати операції користувачів просто так, беручи якусь одну з них. Він намагається застосувати обидві, а вже потім визначає їх пріоритетність роботи.

2.2. Порівняльний аналіз операціональнихтехнологій

Існує ряд технологій, які мають подібну до операціональних трансформацій структуру та особливості, проте відрізняються своєю специфікою роботи з даними і застосуванням. Найпопулярніші з них, про які варто згадати, це Git та CRDT.

Git – програмне забезпечення для контролю версій, яке використовується для роботи з файлами. Порівнюючи Git та систему на основі операціональних трансформацій, можна виділити кілька принципових відмінностей. Git здатен працювати лише з файлами, натомість ОТ технологія дозволяє оброблювати будь-які користувацькі типи даних, визначені програмою. Система автоматичного рішення конфліктів Git вирішує

конфлікти даних сама, а для системи операціональних трансформацій можна самому програмно задати алгоритм правильного злиття даних або ж надати можливість користувачеві самому обрати потрібний варіант злиття даних після конфлікту. Автоматичне рішення конфліктів за допомогою Git системи може бути не завжди таке, як потрібно, адже при виконанні операцій злиття файлів не відомі алгоритми, які були прописані розробниками. Git дозволяє як локально оновлювати файли, так і віддалено, зберігаючи цілісність передачі даних. В своїй базі даних система Git зберігає інформацію та поточний стан файлів у вигляді так званих комітів, де кожен коміт має свій хеш-код. Доступ до файлів здійснюється по їх хешам. На відміну від інших систем контролю версій, Git не зберігає інформацію як список патчів до файлів [12]. Усі дані з їх версіями представляються сукупністю зліпків.

CRDT (Conflict-FreeReplicatedDataType) – технологія на основі типів даних, які можна скопіювати на багато мережевих вузлів і оновлювати паралельно без координації між вузлами. Дана технологія дуже подібна до технології операціональних трансформацій, але на відміну від них не зберігає попередню історію операцій при роботі з вебдодаткам чи іншими застосунками. Як і система операцінальних трансформацій, CRDT система має безліч реплікованих даних на репозиторіях різних користувачів. Кожному користувачу в мережі відповідає певний репозиторій зберігання даних. Останні 10 років CRDT системи використовувалися набагато частіше за OT через наявність повноцінної документації, яка описує інтерфейс роботи з даними в такій системі. По швидкодії та доступності дана система не поступається OT системі, але має принципову відмінність при роботі зі збереженням даних на сервер та їх подальшим злиттям там з метою рішення мережевих інформаційних конфліктів. Відмінність полягає в тому, що при передачі даних в репозиторій у вигляді коміту, система CRDT оперує не сукупністю примінених до даних операцій, як в OT, а поточними станами даних. CRDT модель розроблена таким чином, що вона здатна відслідковувати

стани даних, а потім модифікувати їх згідно отриманих операцій. Після примінення всіх користувацьких операцій до початкових даних отримується поточний стан даних, який для подальших змін буде вважатися за початковий (рис. 2.4). Таким чином в CRDT немає ніякої історії розгалуження даних. Вони просто реплікуються між станами та передаються в поточних станах в свої репозиторії, звідки користувачі їх і дістають. На основі CRDT так само, як і в OT, можна створювати різні лічильники та користувацькі додатки завдяки тому, що технологія підтримує автоматичне рішення конфліктів даних.

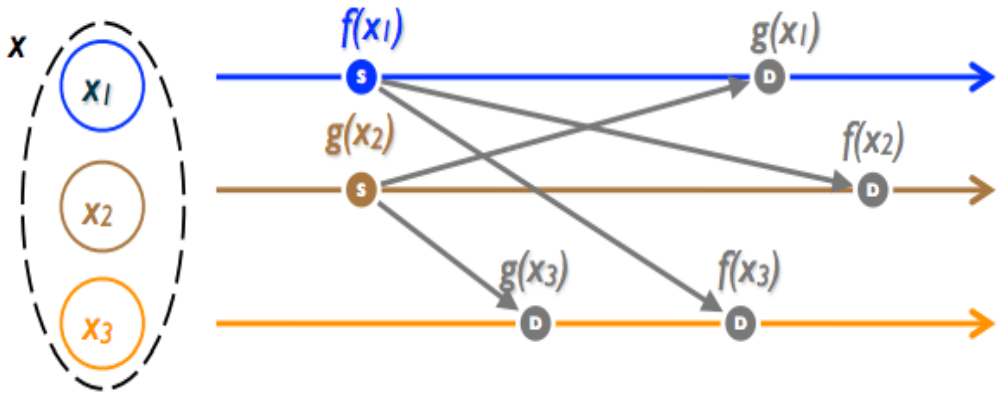

Рисунок 2.4 – Реплікація даних CRDT моделі

Особливу увагу варто приділити новітній технології Global-OT. Дана технологія була розроблена з допомогою мови програмування Java з використанням передового фреймворку DataKernel (він же майбутній ActiveJ). Це технологія розробки розподілених сховищ даних нового покоління в масштабах Інтернету з високими стандартами безпеки даних, відмовостійкості та масштабованості. Вона дозволяє по-іншому розглядати можливості бази даних. Global-OT побудована на основі операціональних трансформацій та пропонує розробникам повний стек інструментів для створення продуктів, які концептуально нагадують Git, але на абсолютно новому рівні. Global-OT реалізує дуже ефективні рішення щодо дизайну мережі та структур даних.

На відміну від багатьох інших технологій, Global-OT дає можливість створювати повністю децентралізовані інтернет-рішення, відкриваючи нові можливості для розробників. Вона має незаперечні переваги перед найбільш популярними технологіями (табл. 2.1).

Таблиця 2.1 – Порівняння характеристик передових Інтернет-технологій для роботи з даними різних типів

	GlobalOT	Global FS	Solid	Git	BitTorrent	IPFS	Block-chain	SQL	NoSQL
Особливості даних									
Будь-які типи даних	✔		✔				✔	✔	✔
Зміна даних	✔		✔	✔		✔	✔	✔	✔
Семантика транзакцій	✔						✔	✔	
Масштабованість									
Застосовується для інтернет-розширеного електронного рішення	✔	✔	✔	✔	✔	✔	✔		
Масштабованість між центрами обробки даних	✔	✔		✔	✔	✔			✔

Продовження таблиці 2.1

Масштабо-ваність в одному центрі обробки даних	✔	✔		✔	✔	✔			✔
Особливості проектування мережі									
Висока відмовостій кість і надмірність в разі помилки	✔	✔		✔	✔	✔	✔		✔
Низька затримка	✔	✔	✔	✔				✔	✔
Робота в оф-лайн режимі	✔	✔			✔	✔	✔		
Збалансування та міграція даних між серверами	✔	✔			✔	✔			✔
Робота з ненадійними серверами	✔	✔			✔	✔	✔		
Особливі можливості									
Аутентифі-каціякрипто гра-фічних даних	✔	✔			✔	✔	✔		

Продовження таблиці 2.1

Багатошаро-векешування	✔	✔			✔	✔	✔		
Шифрування P2Pe	✔	✔				✔	✔		

Global-OT є дійсно однією з технологій майбутнього, адже наразі лише вона здатна уособлювати в собі всі переваги проектування мережі та роботи з різними типами даних. Дана технологія повністю зберігає усі особливості, переваги і недоліки самої технології операціональних трансформацій у контексті розподілених систем роботи з даними. Доступність, передова швидкодія та безпека є її ключовими перевагами. Вона уособлює в собі семантику транзакцій, зміну даних, роботу з будь-якими користувацькими типами даних, масштабованість між кількома одним або ж багатьма центрами керування даними, низьку затримку, високу відмовостійкість та загалом надійну роботу з будь-якими серверами. Потужна система операціональних трансформацій дозволяє не просто використовувати користувацькі типи даних, а й об’єднувати їх в інші, більш складні типи даних. Недоліками даної технології є складність використання та роботи з OT-серверами при розробці вебдодатків. Користувач не може просто так звертатись до серверу та передавати йому дані різних типів, адже на сервері усі дані зберігаються в спеціальних складних графах, які поєднані історією між собою[13]. Технологія Global-OT на даний момент знаходиться в передрелізному стані і не має чіткої документації використання для створення користувацьких додатків. Але навіть не дивлячись на це, розробники надають доступ для використання їх сховищ, побудованих на технології операціональних трансформацій.

2.3. Аналіз властивостей операціональних перетворень

Для кожної системи операціональних трансформацій визначені однакові властивості над операціями, які дозволяють правильно реалізувати перетворення над даними згідно логіки операції. Зазвичай, ці властивості передбачають реалізацію певних функцій перетворення або ж обслуговуються самими алгоритмами операціональних трансформацій.

Існує 3 основні властивості перетворень, які слід враховувати при побудові системи операціональних трансформацій та при реалізації алгоритмів перетворення даних[14].

Перша і основна властивість операціональних перетворень це властивість конвергенції (власне трансформації). Як було зазначено раніше, щоб перевести дані з одного стану в інший, необхідно застосувати до них операції. Операції до даних застосовуються з врахуванням змін від усіх користувачів, щоб якась з операцій не була проігнорована в ході зміни стану даних. Якщо обидва користувачі додатків застосували до даних однакові операції, то жодна трансформація не буде застосована до даних (рис. 2.5).

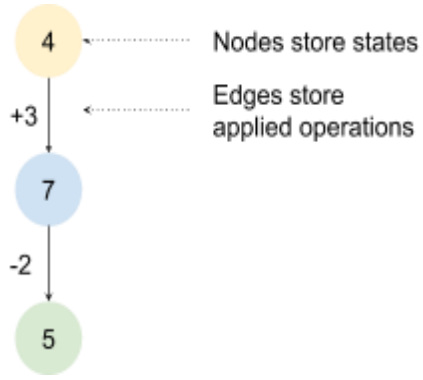

Рисунок 2.5 – Перехід даних з одного стану в інший після застосування операцій

Якщо при відсутності мережі користувачі розійшлися в своїх поточних станах, то при появі мережі злиття їх операцій повинно відбутись правильно. Коли користувачі розійшлися у своїх поточних станах даних (документу, чату, нотаток тощо), то щоб узгодити їх результуючий стан після появи мережі, операції першого користувача застосовуються до поточного стану другого, і навпаки, усі операції другого застосовуються до поточного стану першого користувача (рис. 2.6). Таким чином відбувається операціональне перетворення, а точніше – трансформація операцій, застосованих до даних.

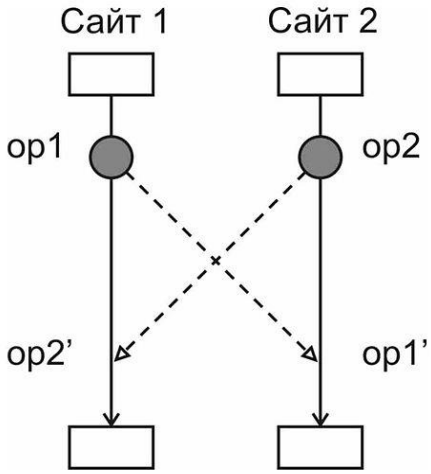

Рисунок 2.6 – Застосування трансформації двох операцій

Властивість конвергенції можна описати наступною формулою (1).

$$op_1 \circ T(op_2, op_1) \equiv op_2 \circ T(op_1, op_2)\,, \qquad (1)$$

де $op_i \circ op_j$ - послідовність операцій, що містить операцію op_i , яка застосовується відразу після op_j;

$\equiv$ – означає еквівалентність двох послідовних операцій;

T – функція операціонального перетворення.

Такі перетворення використовуються тільки в тому випадку, коли система операціональних трансформацій розходиться у своїх поточних станах даних. Рівність (1) також виконується і для більшої кількості операцій (рис. 2.7).

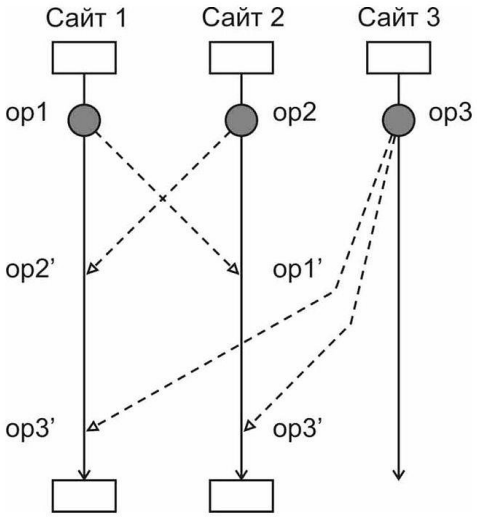

Рисунок 2.7 – Застосування трансформації трьох незалежних операцій

Наприклад для трьох послідовних операцій op_1, op_2 та op_3 формула (1) трансформації набуваєвигляду:

$$T(op_3, op_1 \circ T(op_2, op_1)) \equiv T(op_3, op_2 \circ T(op_1, op_2)). \quad (2)$$

Наступною важливою властивістю операціональних трансформацій є інверсія. Дана властивість полягає в інверсії дії операції, яка застосовується до даних. Наприклад, якщо до даних була застосована операція +1, то інверсія цієї операції буде -1. Важливою ознакою інверсії є те, що застосувавши дану властивість до поточного стану даних, отримується попередній стан (рис. 2.8).

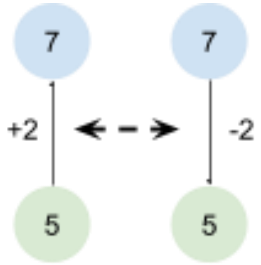

Рисунок 2.8 – Властивість інверсії операцій

Інверсія має 3 підвластивості, які розробники системи операціональних трансформацій застосовують при написанні алгоритмів перетворень операцій. Перша підвластивість описується формулою інверсії операцій (3).

$$S \circ op \circ \overline{op} = S\,, \tag{4}$$

де S – поточний стан даних перед виконання операції;

op – звичайна операція;

$\overline{op}$ – інвертована операція.

Формула (3) показує, що послідовність звичайної та інвертованої операції ідентична пустій операції, після якої стан даних не зміниться.

Друга підвластивість інверсії характеризує те, що послідовність звичайної та інвертованої операцій не впливає на перетворення інших операцій та описується формулою (4).

$$T(op_x, op \circ \overline{op}) = op_x\,, \tag{4}$$

де op – інша операція користувача, яка приміняється до одних і тих самих даних, що й поточна операція;

op_x - поточна операція, що застосовується до даних.

Третя підвластивістьінверсії описується формулою (5) і означає, що інверсія результуючого перетворення операцій буде складатися із послідовного застосування всіх операціональних перетворень до інвертованих операцій, згідно алгоритмів перетворень.

$$\overline{op_1} = T(\overline{op_1},\ T(op_2, op_1)) \text{ та } \overline{op_1} = \overline{T(op_1, op_2)} \quad (5)$$

де $\overline{op_1}$– результат інверсії перетвореної операції;

T – операціональне перетворення до даних.

Останньою ґрунтовною властивістю системи операціональних трансформацій є властивість склеювання, або ж об’єднання. Така властивість дуже подібна до операції gitsquash в системі контролю версій Git. Якщо користувач застосовує кілька операцій до даних, то вони можуть бути об’єднані в одну спільну операцію (рис. 2.9)для підвищення ефективності користування серверним графом станів даних (графом комітів в репозиторії користувача). Об’єднання кількох операцій в одну переводить дані в той самий стан, що і послідовне застосування всіх операцій.

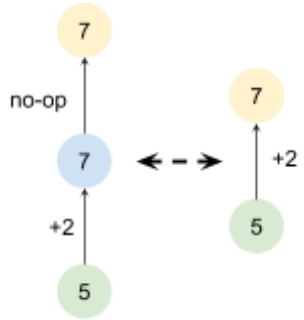

Рисунок 2.9 – Об’єднання двох операцій в одну

2.4. Структура системи операціональних трансформацій

Система операціональних трансформацій будується на сучасних технологіях роботи з мережею та серверами, і може складатися з кількох компонентів. Зазвичай є кілька підходів при плануванні архітектури системи операціональних трансформацій. Найпопулярніший з них – це відокремлення високорівневих алгоритмів трансформації операцій від функцій перетворення низького рівня (рис. 2.10). ОТ система повинна відповідати всім вимогам щодо проектування повноцінної системи і збереження цілісності всіх компонентів. Така система має зберігати всі властивості операціональних трансформацій, які було відмічено в попередніх розділах.

Перш за все, при проектуванні такої системи, обов’язково повинні зберегтись властивості каузальності та допустимості, які будуть застосовувати безпосередньо алгоритми операціональних трансформацій. При операціональних перетвореннях повинен зберігатися порядок застосування перетворень [15]. Будь-який алгоритм в системі операціональних трансформацій вважається коректним, якщо він зберігає властивості каузальності та допустимості. При реалізації таких алгоритмів система буде надійно виконувати свої функції, буде цілісною та буде забезпечувати коректну роботу з даними через операціональні перетворення.

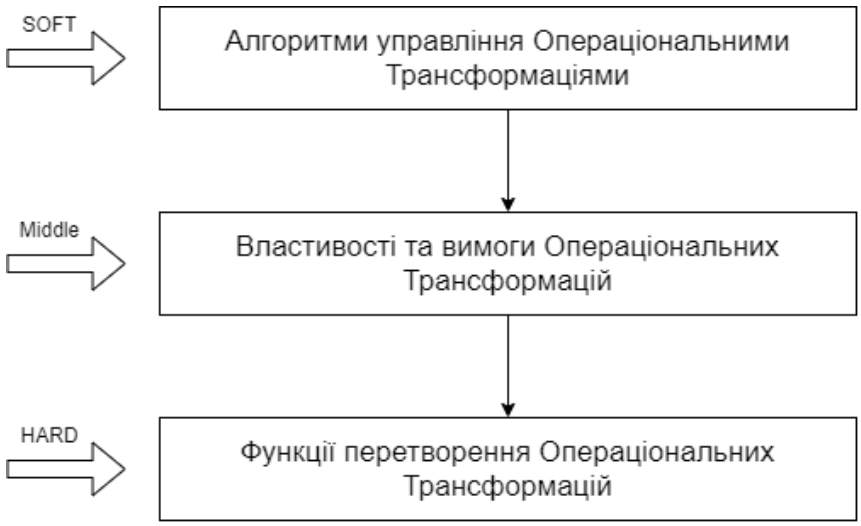

Рисунок 2.10 – Загальна структура системи операціональних трансформацій

На низькому рівні проектування використовуються функції перетворення, які мають задовольняти вимогам звичайного операціонального перетворення та оберненого операціонального перетворення. Останнє полягає в сумісному застосуванні властивості інверсії та об'єднання при переході між станами даних [16].

Кожна система операціональних трансформацій обов'язково повинна мати дві основні моделі – модель даних та модель операцій. Модель даних визначає типи даних, між якими можуть виконуватись операціональні трансформації. Модель операцій визначає, які операції можуть виконуватись над даними, в якому порядку та з якою пріоритетністю. Чим простіші операції можна виконувати над даними, тим простішою буде сама модель даних. Наприклад, якщо для додатку сумісного редагування простих онлайн-документів будуть визначені лише операції додавання та віднімання символів, то модель даних може представляти собою лінійний адресний простір.

2.5. Висновки

Технологія операціональних трансформацій дозволяє вирішувати конфлікти даних будь-якого типу в користувацьких додатках. При роботі з даною технологією використовуєтьсясистема операціональних трансформацій, яка застосовує до користувацьких операцій основні властивості операціональних трансформацій – конвергенцію, інверсію та об’єднання. ОТ технологія має значну перевагу над іншими подібними технологіями в швидкодії, безпеці та надійності при роботі з даними.

3. СТРУКТУРА ТА ОПИС РОБОТИ МОДУЛІВ ОТ ЯДРА НА КЛІЄНТСЬКІЙ СТОРОНІ ВЕБДОДАТКІВ

3.1. Принципи роботи системи операціональних трансформацій

При розробці будь-якого вебдодатку треба наперед ретельно продумувати всю його архітектуру та взаємодію клієнтської та серверної частин. Особливо це стосується розробки додатків, які будуть використовувати технологію операціональних трансформацій та взаємодіяти з ОТ системою. Використовувати технологію операціональних трансформацій можна було б для багатьох додатків, проте через складну побудову віддалених ОТ серверів додатки не мають змогу обмінюватися даними між собою в зручний спосіб. При потребі зробити запит на віддалений сервер дані доводиться щоразу трансформувати та розподіляти в інші структури даних, адже віддалений сервер зберігає всі дані у вигляді складного графа з комітами, який має незмінну історію. Аналогічною проблемою є і зворотній запит до репозиторію, адже дані, що повертаються з серверу надходять в додаток не в тих типах даних, які потрібні для коректного їх відображення користувачу. Саме тому постає необхідність в розробці спеціального програмного забезпечення, яке змогло б працювати з шаблонними типами даних на стороні клієнта.

Відомо, що на стороні серверу усі запити надходять до так званої серверної OT-Node (ОТ ядро), яка представляє собою складну систему обробки запитів та реалізує всі необхідні алгоритми операціональних перетворень з даними, які знаходяться в комітах графа. Побудова такого самого ОТ ядра, але вже на стороні клієнту, а не сервера, значно б спростила інтеграцію технології операціональних трансформацій в будь-які вебдодатки.

Щоб побудувати клієнтське програмне забезпечення для взаємодії з OT-серверами, необхідно детально проаналізувати принципи роботи OT серверу та структуру серверного OTядра.Для дослідження серверної частини OT розглянемо технологію Global-OT, яка наразі є передовою в сфері швидкодії та безпеки доступу до даних.

Серверна частина OT - це повністю децентралізована багаторівнева мережа без єдиної точки відмови та ефективного проектування. Її мережа складається з клієнтського рівня, двох шарів P2P OT-серверів - Caching і Master та шару незалежних центрів управління - DiscoveryServices [17].

Клієнтські програми повинні використовувати OT-Driver - спеціальний драйвер, який забезпечує взаємодію між додатками та серверами. Екземпляри драйвера OT не передають дані між собою безпосередньо.

Всі OT сервери є взаємозамінними та застосувальними, тому один сервер може працювати з довільними додатками OT. OT-Server може бути Master (якщо зберігає оригінальне сховище), Caching (якщо зберігає кеш сховища) або комбінувати обидві ролі. Кешування також забезпечує попереднє завантаження. Найчастіше запитувані сховища автоматично кешуються на OT-серверах, значно збільшуючи швидкість доступу до даних. OT-сервери взаємодіють із серверами спеціального сервісу(DiscoveryService).

Для оптимізації користувацького досвіду та збільшення швидкості доступу до даних, Global-OT має декілька центрів контролю - DiscoveryServices. Вони виконують роль, схожу на DNS, збираючи інформацію про те, де і в яких сховищах зберігаються, і надають ці дані користувачам. Також служби Discovery перевіряють дійсність щойно завантажених сховищ за допомогою їх цифрових підписів.Завдяки цифровим підписам вся система має надійний рівень безпеки доступу до даних. Також Global-OT дозволяє аутентифікацію користувачів - їхні відкриті ключі можуть використовуватися як унікальний логін. Таким чином, власники сховищ можуть створювати власні списки надійних відкритих ключів або додавати їх

у чорні списки. Крім того, користувачі отримують інструменти управління довірою - вони можуть працювати зі сховищами, яким вони довіряють, фільтруючи дані, які надходять не з надійних джерел.

OT технологія може працювати з довільними типами даних та операціями, визначеними користувачем. Всі дані, що зберігаються в Global-OT, надійно захищені від будь-яких можливих сторонніх захисників даних завдяки багатошаровим технологіям безпеки. Global-OT організовує всі дані як сховища. У кожному сховищі є пара унікальних публічних та приватних ключів. Публічний ключ використовується як ідентифікатор, а приватний ключ використовується для цифрових підписів, які автоматично додаються до кожного коміту. Таким чином, лише власник сховища може створювати коміти, а дійсністькомітів може перевіряти кожен, хто звертається до них [18].

Більше того, кожен коміт може бути додатково зашифрований за допомогою самогенерованого симетричного ключа. Лише кінцеві користувачі, які мають ключ, можуть розшифровувати коміти. Тим не менш, підписи можуть бути перевірені будь-яким проміжним сервером Global-OT, який працює зі сховищем, навіть якщо він не забезпечений ключем шифрування. Це дозволяє перевірити справжність комітів для безпеки кешування та попереднього завантаження.Користувачі можуть створювати копії зі сховищ власника та застосовувати деякі модифікації, які не вплинуть на оригінальне сховище.

Global-OT дозволяє репозиторіям підписатися на зміни один одного, щоб сховища могли синхронізуватися без офіційних запитів на оновлення. Таким чином можна просто створити розповсюджені месенджери. Global-OT може використовуватися як основна база даних програми. Такий підхід дозволяє створювати вперше розповсюджені офлайн-програми з простою та ефективною синхронізацією між користувачами.

Global-OT - це врешті-решт послідовна система, тому вона може бути використана як дерево LSM (дерево-структуроване злиття) для систем з OLAP

або OLTP. Додатки без використання операціональних трансформацій несинхронізовано збирають велику кількість інформації з різних джерел, тоді як Global-OT забезпечує синхронізацію та узгодженість даних на всіх користувацьких комп'ютерах.

3.2. Вибір інструментального програмного забезпечення

Вибір інструментального програмного забезпечення є однією з важливих частин при розробці будь-якого додатку або іншого програмного застосунку. На сьогоднішній день існує багато надійних та зручних інструментів для розробки програмного забезпечення будь-якого типу.

При виборі інструментального програмного забезпечення необхідно врахувати, які мови програмування найкраще підходять при реалізації клієнт-серверної архітектурі додатків. Віддалений OT сервер з усіма його алгоритмами операціональних трансформацій написаний мовою програмування Java, адже вона має певні можливості в області шифрування даних, що необхідно для забезпечення безпеки доступу в додатках та безпеки усіх даних в цілому. Для взаємодії з сервером і надсилання клієнтських запитів найбільш зручною мовою програмування є мова Javascript. Вона є найпопулярнішою мовою програмування на даний момент на ринку інформаційних технологій, а також має велику кількість фреймворків для швидкого створення вебдодатків та інших клієнтських застосунків. МовоюJavascript зручно створювати різні шаблонні інтерфейси функцій для роботи з даними. Саме це і потрібно використати при створенні клієнтської OT-Node.

Окрім створення клієнтської OT-Node, яка буде працювати з OT системою та віддаленим OT сервером, важливим є процес розробки додатку,

який буде використовувати це програмне забезпечення для посилання запитів на ОТ сервер. Для написання цього додатку використовується бібліотека React.js, яка зараз є найбільш популярною серед веброзробників на мові Javascript.

React.js – це надзвичайно сучасна відкрита для доступу бібліотека, яка використовується розробниками для створення різноманітних інтерфейсів користувача у вебдодатках. React.js забезпечує малювання візуальних компонентів, які будуть показані користувачеві в додатку. При цьому бібліотека вирішує проблеми з оновленням даних на вебсторінці та дозволяє зберігати дані, які будуть змінюватися в проміжних змінних різних класів, які звуться станами даних. Для малювання компонентів інтерфейсу в додатку використовуєтьсябібліотека дизайну –MaterialUI. Вона призначена для гарного візуального оформлення елементів сторінки та має гарну інтеграцію з бібліотекою React.js.

Специфіка роботи з конкретною мовою програмування може залежати не лише від самої мови програмування або її бібліотек чи фреймоворків. Важливим є також і вибір середовища програмування даною мовою, оскільки різні середовища програмування мають різний набір інструментів та свої особливості для роботи. Для розробників веб додатків клієнт-серверної архітектури нині є передові середовища програмування, які надає фірма JetBrains. Для мов програмування Java та Javascript такими середовищами є IntelijIdea та WebStorm відповідно. Для локального запуску серверів мовоюJava використовується IntelijIdea з вбудованим фреймворком до Java – Maven.

Для створення вебдодатку та клієнтського вузла на основі операціональних трансформацій (OT-Node) використовуєтьсясередоващеWebStorm. Воно ідеально підходить для будь-якого веброзробника, містить весь необхідний інструментарій та вбудовану інтеграцію з усіма сучасними фреймворками та бібліотеками мови Javascript.

3.3. Опис структури ОТядра на клієнтській частині

Клієнтське ОТядро представляє собою програмне забезпечення для роботи з системою операціональних трансформацій, яке дозволяє робити серверні запити та працювати з будь-якими типами даних. По своїй суті клієнтська ОТ нода є інтерфейсом, розробленим мовоюJavascript, який дозволяє в швидкий та зручний спосіб взаємодіяти з ОТ сервером. Дане програмне забезпечення має високий рівень інкапсуляції для користувача, адже воно дозволяє безперешкодно робити запити з операціями прямо на ОТ сервер без розуміння того, що на сервері існують операціональні графи та складні алгоритми злиття та перетворень даних.

Як уже зазначалось раніше, кожен додаток взаємодіє зі своїм власним репозиторієм, до якого поступають коміти (запити з даними від користувачів прямо з вебдодатку). Щоб побудувати ОТ ядро на клієнті, недостатньо знати лише властивості системи операціональних трансформацій. Сама ж функціональная частина ОТ системи – це лише програмний модуль, який дозволяє правильно налаштовувати дані та операції на подальшу роботу з ними на сервері. Клієнтська ОТ нода містить ряд запитів для взаємодії з сервером. Кожен запит взаємодіє з комітами, які робляться в репозиторій, тому важливо детально розглянути структуру комітів, а також усі запити, які з ними взаємодіють.

Кожний коміт, подібно до системи Git, містить певну інформацію, необхідну для збереження даних в репозиторії та подальшої обробки даних системою операціональних трансформацій на сервері [19]. Коміт містить ідентифікатор репозиторія, час створення коміту, ідентифікатор батьківсьогокоміту та масив операцій, які примінялись до поточного стану даних (рис. 3.1). Ідентифікатор репозиторію є обов’язковим, адже саме так користувач зв’язує свої запити зі своїм існуючим сховищем даних. Час створення коміту дозволяє ОТ серверу правильно застосовувати

операціональні перетворення до даних, у разі, якщо всі застосовані операції були рівні між собою. Також час створення коміту допомагає формувати правильну історію редагування даних у графі ОТ серверу.

Рисунок 3.1 – Структура коміту в системі операціональних трансформацій

Крім зазначених компонентів, структура коміту може мати й інші, які безпосередньо стосуються даних, або ж налаштування самого сховища. Такі компоненти є опціональними і на практиці використовуються досить рідко. Проте найважливішими складовими коміта є батьківський id, який містить в собі хеш-код попереднього коміта у графі, та список примінених до даних операцій. Всі операції зберігаються в масиві, причому кожна операція передається в серіалізованому форматі. Це необхідно для коректної передачі типів операцій через мережу.

Як відомо, найчастіше дані передаються в форматі JSON, проте граф на ОТ сервері не сприймав би такий формат даних, як JSON, адже йому необхідно передавати дані в сирому вигляді (бітове та байтове представлення). З ОТ серверу надходять коміти, які зберігаються в форматі BLOB, який в свою чергу являє собою бінарний набір даних, що розглядається як єдиний, неперервний об’єкт. Саме через це клієнтська частина має містити 2

спеціальні перетворювачі даних, які називаються серіалайзери. Вони здатні не лише перетворювати дані формату JSON та серіалізувати строки даних з вебдодатку, а й працювати з комітами типу BLOB. Коміти даних перетворюються в комплексні дані типу BLOB, після чого відправляються на OT сервер, де серверне програмне забзпечення почергово додає коміт в граф, де всі дані уже представлені у вигляді бінарного набору в єдиному об'єктікоміта.

OT ядро на клієнтській частині повинно включати в себе набір певних певних компонентів, які взаємопов'язані між собою і разом будуть виконувати основну задачу ядра – взаємодію з віддаленим OT сервером. Ці компоненти представляють собою класові інтерфейси, з якими користувачу буде зручно взаємодіяти прямо з вебдодатка.

Враховуючи усі ці особливості та беручи до уваги моделі, які є обов'язковими для системи операціональних трансформацій (а саме модель даних та операціональна модель), можна виділити чотири основні структурні блоки клієнтської OT-Node. Ними є 2 класи, що відносяться до моделі даних – це інтерфейс класу-коміту (OTCommit) та інтерфейс класу-менеджеру роботи з даними (OTStateManager), та інші 2 класи, які представляють операціональну модель, і стосуються обробки даних і взаємодії з OT сервером. Цими двома класами є безпосередньо головний клас самої OT ноди (OTClientNode), який оброблює всі запити на сервер, та клас для роботи з перетворенням операцій, який містить математичні функції трансформації операцій, базуючись на властивостях системи операціональних трансформацій. Останні два класи є набагато складнішими за перші два в реалізації, оскільки на них покладена основна функція OT ядра: вони є проміжною ланкою між користувацьким додатком, з якого надходять запити в це ядро, та OT сервером, куди саме ядро має перенаправляти ці запити, попередньо перетворюючи дані з операціями.

3.4 Розробка модулів програмного забезпечення OT

Програмне забезпечення на основі операціональних трансформацій, яке представляє собою клієнтське ядро з системою операціональних трансформацій, містить 4 основні модулі – OT-System, OTCommit, OTClientNode та OTStateManager.

OT-System є однією з головних складових частин клієнтського ядра операціональних трансформацій. Вона відповідає за різні математичні обчислення та містить набір функцій, які трансформують операції над даними (рис. 3.2). Ці функції реалізують усі основні властивості операціональних перетворень, які були описані раніше, а саме: конвергенція (трансформація операцій), інверсія операція та об'єднання (склеювання).

```
class OTSystem<S> {
  _transformers: Map<Class<OTOperation<S>>, Map<Class<OTOperation<S>>, transformer<S>>>;
  _squashers: Map<Class<OTOperation<S>>, Map<Class<OTOperation<S>>, squasher<S>>>;
  _emptyPredicates: Map<Class<OTOperation<S>>, emptyPredicate<S>>;
  _inverters: Map<Class<OTOperation<S>>, inverter<S>>;

  constructor(transformers: Map<Class<OTOperation<S>>, Map<Class<OTOperation<S>>, transformer<S>>>,
              squashers: Map<Class<OTOperation<S>>, Map<Class<OTOperation<S>>, squasher<S>>>,
              emptyPredicates: Map<Class<OTOperation<S>>, emptyPredicate<S>>,
              inverters: Map<Class<OTOperation<S>>, inverter<S>>) {
    this._transformers = transformers;
    this._squashers = squashers;
    this._emptyPredicates = emptyPredicates;
    this._inverters = inverters;
  }
```

Рисунок 3.2 – Реалізація інтерфейсу OTSystem.

Операція над даними має шаблонний тип, адже вона може бути будь-якою, щоб задовольняти вимогам універсальності застосування клієнтського ядра для будь-яких додатків. Для побудови OTSystem використовується спеціальний білдер класу та додаткові утиліти для роботи зі сформованим результатом операцій. Крім функцій інвертування, трансформації та об'єднання операцій (squash), OTSystem містить ще одну додаткову операцію

isEmpty, яка перевіряє, чи отримана операція не є пустою. Це необхідно для того, щоб оптимізувати процес взаємодії з сервером, та не застосовувати операцію до поточного стану даних, якщо вона пуста. Таким чином буде підвищена швидкодія відклику на запити у вебдодатку. Пуста операція може виникнути після застосування кількох послідовних операцій або виконання навмисного інвертування. Саме тому дану функцію слід застосовувати і одразу при отриманні масиву операцій, і перед відправкою їх на сервер, коли, наприклад, після виконання функції об'єднання операцій сумарна операція над даними може виявитись пустою.

Другим класом клієнтської ОТ ноди є інтерфейс класу OTCommit. Цей клас необхідний для створення структури коміту, яка буде відправлена на ОТ сервер. OTClientNode та OTStateManager взаємодіють з цим класом, коли обробляють запити на сервер. В ході реалізації використовуються додаткові утиліти мовоюJavascript для створення правильної структури коміту та перетворення JSON масиву операцій у формат BLOB. OTCommit реалізує функції послідовного створення структури коміту для подальшої його обробки. Структура коміту в інтерфейсі класу є такою, як описана в попередньому розділі.

Для зберігання даних та оновлення їх стану використовується третій модуль – OTStateManager. Він містить набір функцій для роботи з даними, які приходять від користувача. Цей модуль взаємодіє з модулем комітів та модулем запитів на ОТ сервер (рис. 3.3). Він містить відомості про хеші поточного та батьківського комітів, які використовуються для ідентифікації вузлів на серверній частині OT-Node, де всі коміти по суті є цими вузлами зберігання даних у графі. В даного класу є 2 найголовніші методи, без яких він не зміг би використовуватись іншими модулями. Це методи apply() та getState(). Метод apply() застосовує операції, які були визначені користувачем, і прийшли в запиті до поточних даних в батьківському коміті. На основі цього формується новий коміт (новий зліпок даних уже із застосованими

операціями). Метод getState() повертає поточний стан даних до або після застосування операцій, залежно від місця виклику в програмі.

```
export class OTStateManager<TKey, TState> {
  _initState: provider<TState>;
  _otNode: OTNode<TKey, TState>;
  _otSystem: OTSystem<TState>;
  _retryTimeout: number;

  _state: TState;
  _revision: TKey | null = null;
  _level: number | null = null;
  _workingOperations: Array<OTOperation<TState>> = [];
  _pendingCommit: Blob | null = null;
  _eventEmitter: EventEmitter = new EventEmitter();
  _stopPolling: null | () => void = null;
  _retryTimeoutId: TimeoutID | null = null;

  constructor(
    initState: provider<TState>,
    node: OTNode<TKey, TState>,
    otSystem: OTSystem<TState>,
    retryTimeout: number = DEFAULT_RETRY_TIMEOUT
  ) {
    this._initState = initState;
    this._otNode = node;
    this._otSystem = otSystem;
    this._state = this._initState();
    this._retryTimeout = retryTimeout;
  }
```

Рисунок 3.3 – Реалізація інтерфейсу OTStateManager

OTStateManager містить додатковий приватний метод pull. Він є одним з ключових при внутрішній роботі з комітами та їх обробці. Сам метод працює подібно до методу pull в системі Git (рис. 3.4). Він намагається об’єднати локальну версію даних з тією версією, яка прийшла з серверу після виконання запиту отримання даних (fetch).

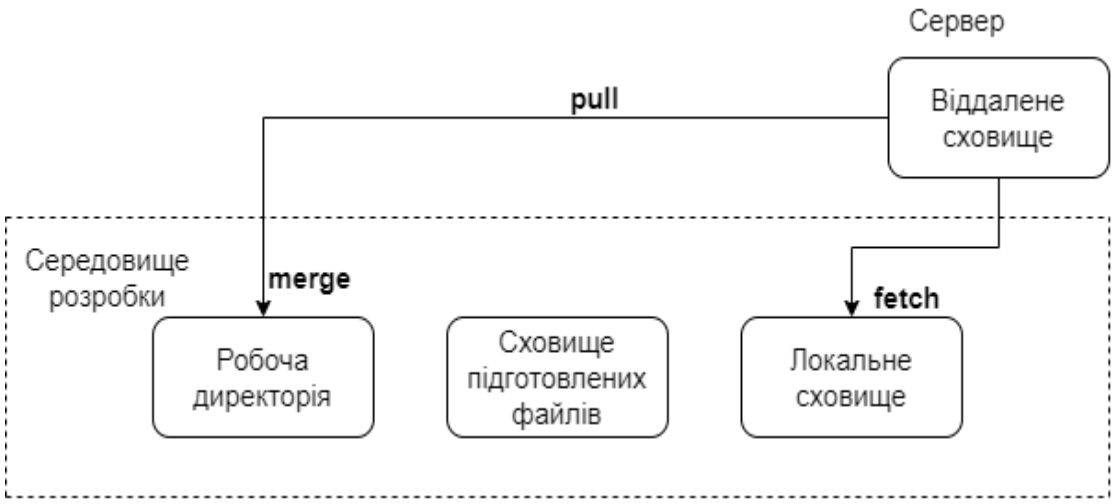

Рисунок 3.4 – Демонстрація роботи методу pull в системі Git

Останнім та найскладнішим в реалізації модулем є OTClientNode, тобто власне самий клас реалізації ядра, який містить взаємодію з усіма попередніми модулями, різноманітні серіалізатори даних та включає в себе всі функції обробки запитів. Далі представлено запити, які можуть надсилатися на OT сервер, та яку логіку вони уособлюють.

Усі запити працюють з комітами, подібно до системи Git та реалізовані за допомогою сучасних технологій асинхронності в Javascript, а саме async/await. Більшість із запитів мають такі ж назви, як і в Git, оскільки самі моделі систем надзвичайно схожі.

Запити, що реалізує клієнтське OT ядро до серверу для підтримки взаємодії вебдодатку та OT серверу:

- createCommit
- push
- checkout
- poll
- fetch

Метод createCommit забезпечує створення коміту у форматі BLOB для його передачі на сервер. Використовує POST метод запиту клієнт-серверної архітектури та чекає відповіді від серверу за допомогою методу отримання даних fetch. Цей метод має таку саму логіку роботи, як і в системі Git (рис. 3.5), а також повертає в користувацький додаток уже дані правильного формату. Під правильним форматом розуміється формат JSON, оскільки не можна повертати у вебдодаток бінарне необроблене представлення даних типу BLOB.

Рисунок 3.5 – Демонстрація роботи методу fetch в системі Git

Надзвичайно схожим до методу fetch є метод poll. Йому немає альтернатив в інших системах з подібною архітектурою через його єдину особливість. Як і метод fetch, метод poll використовується для отримання даних з серверу та на відміну від fetch, який повертає лише поточну ревізію (поточний стан з коміту), poll здатний зависнути та чекати змін від серверу, а потім відразу їх повертати користувачеві. Таким чином забезпечується синхронізація вебдодатків після виходу з режиму офлайну. Застосування даного метода дозволяє клієнтській частині застосунків весь час бути синхронізованою з сервером, не роблячи додаткових запитів. Як тільки на сервері відбудуться зміни – метод poll відразу поверне клієнту поточну

ревізію і масив змін (рис. 3.6). Метод дозволяє користувачу завжди бути синхронізованим з останніми змінами на ОТ сервері. Для реалізації самого методу в класі OTClientNode використовуються сучасні технології очікування відповіді запиту, побудовані на промісах мови Javascript.

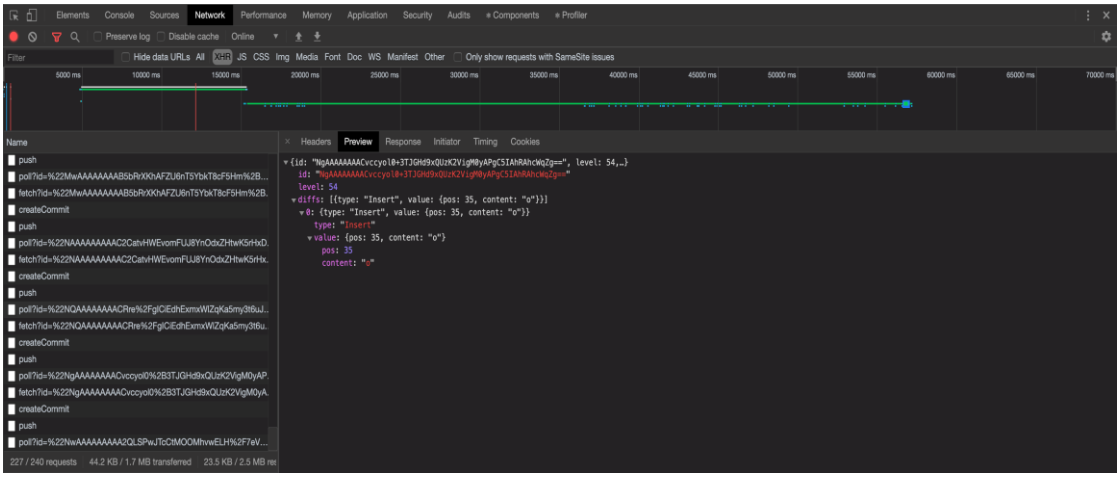

Рисунок 3.6 Запит методу poll в одному з вебдодатків

Метод push приймає коміт, створений за допомогою методу createCommit, відправляє дані у вигляді формату BLOB на ОТ сервер, та повертає на клієнт створений коміт. При формуванні запиту використовується такий самий метод запиту, як і для функції createCommit, а саме метод POST.

При запуску вебдодатку необхідно зробити його початкову ініціалізацію, яка передбачає отримання всіх необхідних даних з серверу, починаючи від кореня графа (або від певного вузла серверного графа) і до поточного стану даних. Для цього використовується запит checkout(), який здатен проініціалізуативебдодаток. Це перший метод, який має викликатися з вебдодатку на ОТ сервер. У вигляді промісу він повертає ревізію (тобто ідентифікатор коміту) та поточний рівень у серверному дереві-графі.

Важливим є те, що програмне забезпечення (клієнтське ОТ ядро) може використовуватись в додатках будь якого типу. Приклад одного з таких додатків наводиться в наступному розділі.

3.5 Опис інтерфейсу вебдодатку на основі ОТ технології

На основі технології операціональних трансформацій можна розроблювати безліч різноманітних вебдодатків та застосунків. Найпопулярніше застосування даної технології набуло саме в розробці колаборативних додатків онлайн-редагування текстових простих документів (наприклад застосунок GoogleWave) [20]. Проте на основі цієї технології можна створити й інші за своєю специфікою додатки. Це можуть бути онлайн-чат, відео-транслятор, додаток для колаборативної обробки зображень, текстові користувацькі нотатки або todo список. Для всіх цих додатків зберігається ідентичність роботи з ОТ серверами та клієнтським ОТ ядром. Більшість з наведених вебдодатків мають багато альтернатив в сучасному світі інформаційних технологій. Саме тому мовоюJavascript був розроблений унікальний користувацький додаток GlobalFiles, який водночас демонструє роботу з операціональними трансформаціями ОТ технології та з простими файлами операційної системи. Варто додати, що такий додаток має надзвичайно високу систему безпеки, яка використовує спеціальні методи шифрування інформації в користувацьких файлах (рис. 3.7). Вона базується на розділенні файлів на декілька частин, кожна з яких представляється окремою вхідною точкою з відповідним їй хешем.

Даний додаток використовує розроблене клієнтське ОТ ядро для взаємодії з вже готовими відкритими Global ОТ серверами. Вхід в додаток здійснюється по спеціальному приватному ключу (рис. 3.8), який генерується випадковим чином на сервері та надається користувачу один раз. Здійснити вхід в додаток можна, завантаживши файл з приватним ключем. Сам ключ представляє собою рядок з випадкових неповторних символів і надається для доступу в свій репозиторій.

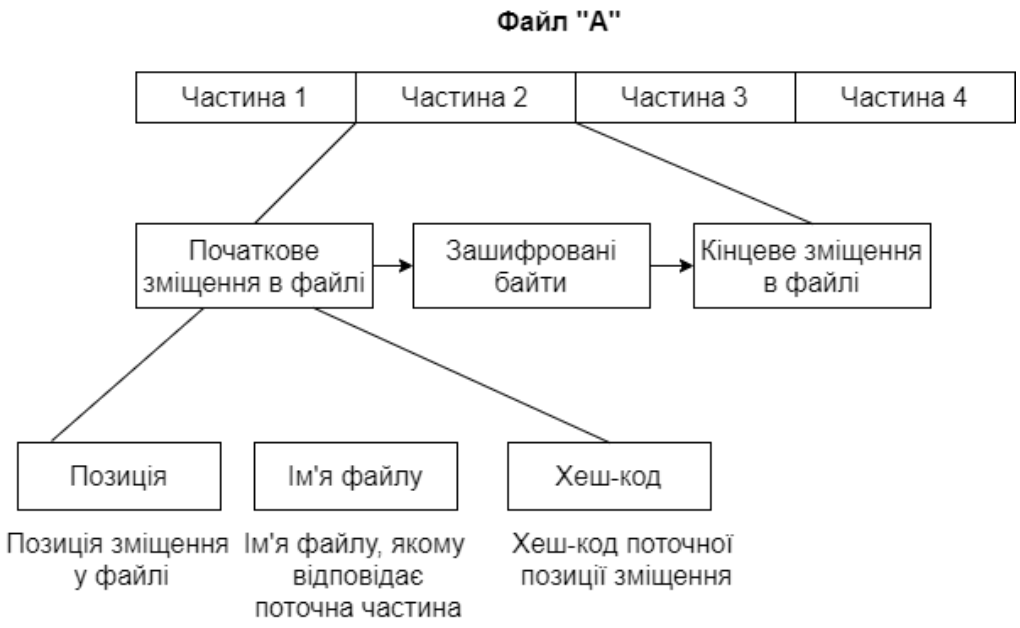

Рисунок 3.7 – Система шифрування файлів в додатку GlobalFiles

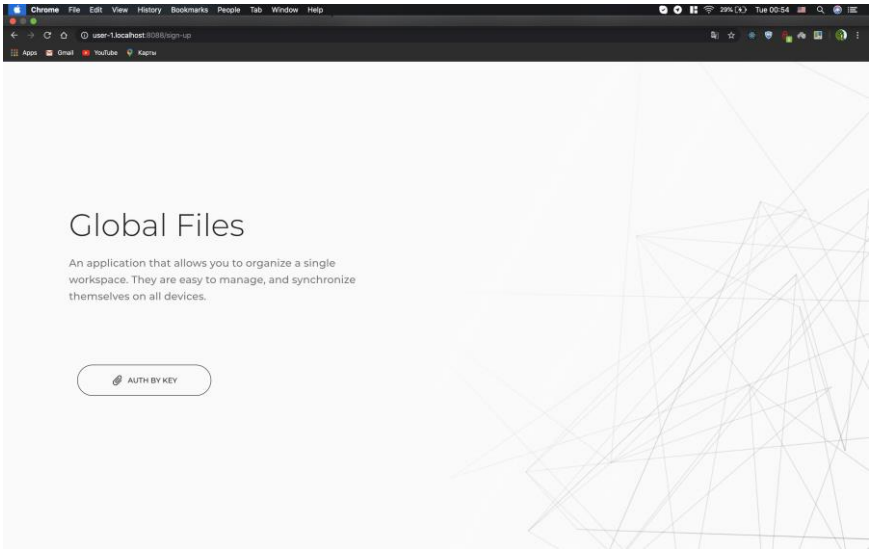

Рисунок 3.8 – Початковий екран входу до додатку через приватний ключ

Додаток GlobalFiles реалізовує повноцінну файлову систему для одного користувача. При цьому, завдяки використанню технології операціональних трансформацій, користувач має змогу безперешкодно створювати та видаляти папки та файли з різних приладів одночасно без виникнення конфліктів даних.

Сам інтерфейс додатку має простий зручний дизайн (рис. 3.9) та дозволяє додавати для збереження та переглядати будь які файли (рис. 3.10).

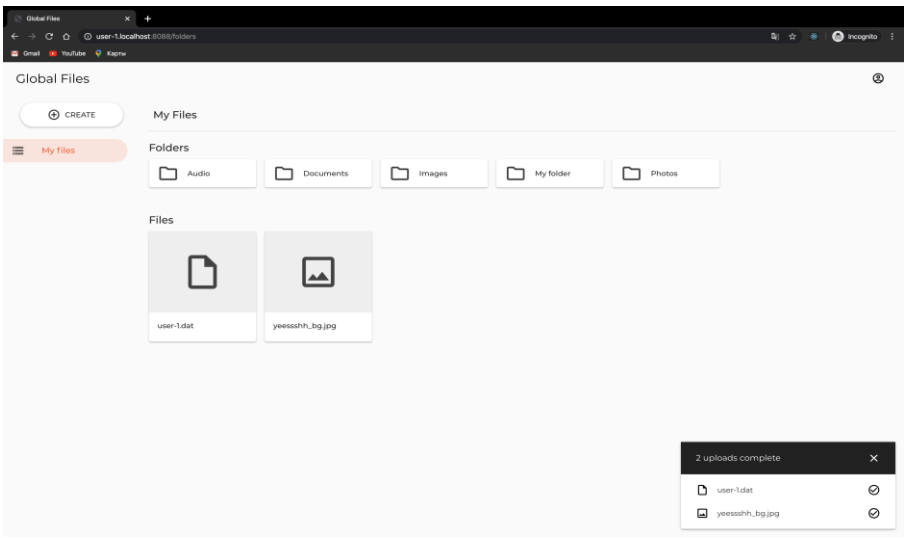

Рисунок 3.9–Інтерфейс додатку GlobalFiles

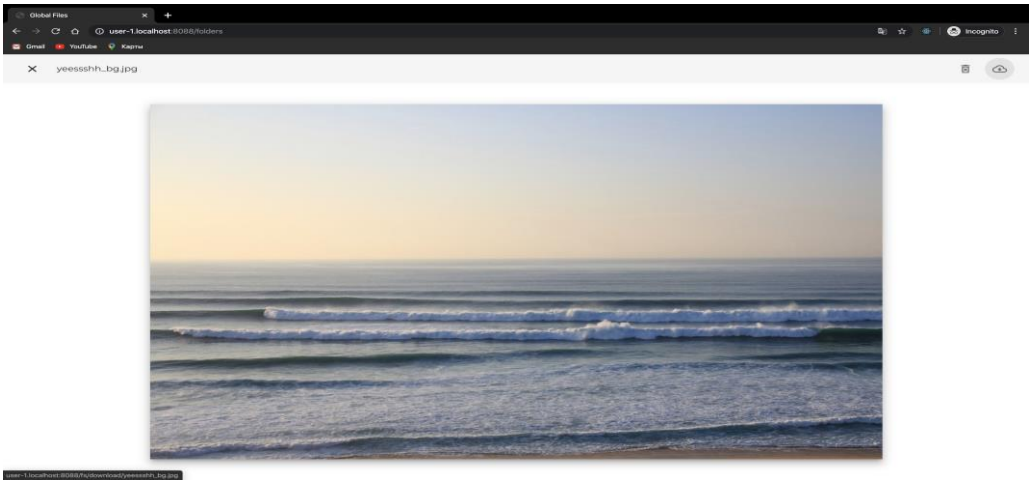

Рисунок 3.10 – Приклад перегляду файлу в додатку GlobalFiles

При реалізації додатка використовувались спеціальні сервіси, сумісні з бібліотекою React.js, які дозволяють робити запити на клієнтське ОТ ядро та через нього отримувати інформацію з серверу (рис. 3.11). Оскільки серверу не відоме поняття директорії, то для їх відображення використовується отримання з серверу штучного файлу ~#!HIDDEN!#~ який і буде визначати

наявність пустої папки. Якщо з серверу надходять вкладені файли, то це уже передбачає створення папки на клієнті. Всі файли, що зберігаються на сервері, можна отримати за допомогою запиту /list, який поверне всю ієрархію файлової системи.

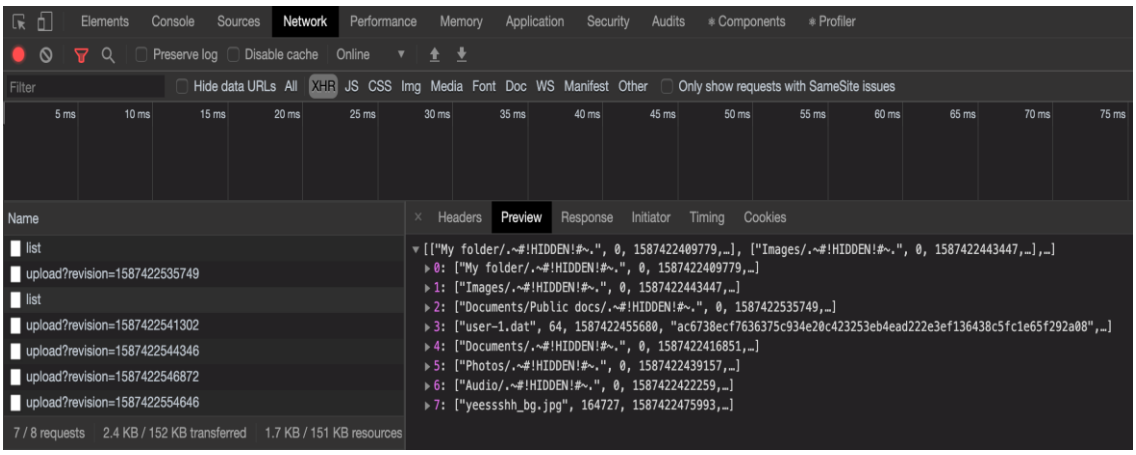

Рисунок 3.11 – Відповідь серверу на запит отримання всіх збережених файлів та директорій (запит /list)

3.6. Висновки

Розроблено вебдодаток файлової системи, який використовує клієнтське програмне забезпечення на основі операціональних трансформацій для взаємодії з віддаленим ОТ сервером. Програмне забезпечення складається з чотирьох основних модулів, які застосовують до даних основні властивості ОТ, а також надсилають запити до сховища збереження ОТ даних на сервері.

4. ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

4.1. Тестування програмного забезпечення

Розробка програмного забезпечення на основі операціональних трансформацій вимагає ретельного тестування всіх його структурних компонентів. Це зумовлено високою складністю побудови архітектури самої системи операціональних трансформацій. При розробці клієнтського ОТ ядра слід написати ряд тестів до всіх його модулів, а також за допомогою тестів перевірити взаємодію компонентів в самих модулях та вивести результат їх взаємодії один з одним. Крім того, оскільки вебдодаток буде використовувати дане програмне забезпечення, то треба протестувати всю роботу розробленого користувацького додатку, а також перевірити запити, що надсилаються до клієнтського ОТ ядра з додатку. Окрему увагу слід приділити розробці тестів для перевірки коректності роботи запитів, які надсилаються від вебдодатку до ОТ ядра та від ОТ ядра до серверу операціональних трансформацій.

Для розроблення модулів тестування клієнтського ОТ ядра мовоюJavascript використовується спеціальний тестувальний модуль jest, який дозволяє одночасно запускати декілька тестів, а також показує розробнику програмного забезпечення усі помилки, які виникли в процесі проведення тестування.

Основна частина тестів націлена на тестування модулів OTSystem та OTClientNode, які виконують головні функції взаємодії між ОТ сервером та вебдодатком, а також забезпечують основні перетворення даних згідно отриманих операцій.

Для реалізації тестування окремих модулів ОТ ядра розроблюються додаткові класи тестування мовоюJavascript. Такі класи будуть перевикористовувати роботу один одного в режимі тестування. Основною

задачею тестування клієнтського OT ядра та вебдодатку є виявлення помилок при роботі з операціональними властивостями в системі операціональних трансформацій та перевірка працездатності клієнтських алгоритмів злиття даних, які забезпечуються цими операціональними трансформаціями. Розроблене тестувальне програмне забезпечення дозволяє вирішити цю задачу, а також наглядно демонструє роботу операціональних перетворень та застосування властивостей OT системи до даних, виводячи в консоль розробника результати тестувань у вигляді міні-графів операціональних трансформацій (рис. 4.1).

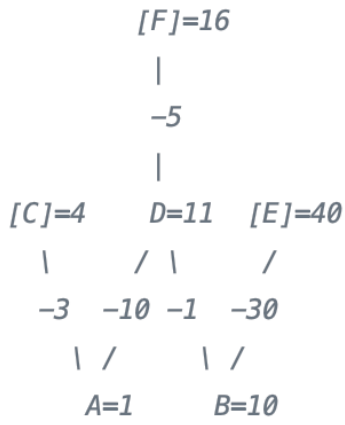

Рисунок 4.1 – Застосування операціональних трансформацій до різних станів даних

Базовими тестувальними методами для перевірки працездатності операціонального злиття даних є такі методи, як TestAddOp, TestOTResolver, TestSetOp та основний метод merge() для застосування до даних операцій від кількох користувачів. Методи TestAddOp та TestSetOp безпосередньо взаємодіють з модулем даних (OTStateManager), який застосовує до утворених операцій метод apply(), після чого всі операції відправляються на OT сервер у вигляді коміту, де й застосовуються до нього. Після застосування операцій до

даних кожен коміт зберігається в серверному графі, а тестове забезпечення дозволяє отримати потрібну частину графа з серверу та вивести його в термінал.

Виведені до терміналу графи зображують процес перетворення операцій та їх застосування до даних під час взаємодії клієнтського ОТ ядра та серверу операціональних трансформацій. Латинськими літерами в даному графі позначаються поточні стани даних, та версії, в яких розійшлися користувачі. Запущені тести заздалегідь описують набір початкових станів даних з їх значеннями, а також почерговий набір операцій, який буде застосовуватися до даних. Важливо, що деякі графи демонструють розпаралелюваннягілок станів даних, які в результаті злиття мають дійти єдиної працюючої версії, або ж єдиного стану даних (рис. 4.2).

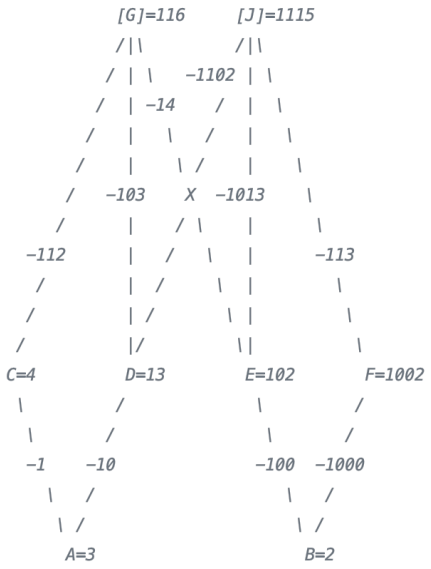

Рисунок 4.2 – Приклад паралельного застосування різних операцій до станів даних

Враховуючи, що є кілька різних станів даних, які завдяки операціональним перетворенням будуть зливатись в один єдиний стан, можна зробити висновок, що усі ці стани відповідають окремим користувачам, які

редагують одні й ті самі дані з вебдодатків. Проте, це може й бути один користувач, який працює з набором даних в режимі офлайн з кількох пристроїв одночасно.

4.2 Аналіз результатів роботи

В ході роботи над дипломним бакалаврським проєктомрозроблено клієнтське програмне забезпечення, яке можуть використовувати будь-які сучасні вебдодатки, що проектуються на основі використання операціональних трансформацій. Створено програмне забезпечення, а саме вебдодатокGlobalFiles, який яке по своїй суті являє цілісну програмну систему з чотирьох комплексних модулів. Додаток є повноцінною файловою системою, що працює в режимі онлайн як сховище файлів та обмінюється даними типу JSON зі створеним програмним забезпеченням. Таке програмне забезпечення на стороні клієнту зветься клієнтським OT ядром, що здатне взаємодіяти з сервером, перетворюючи дані формату JSON в потрібний серверу формат BLOB.

Додатково-розроблене тестувальне програмне забезпечення дозволяє відслідкувати перетворення даних, які відбувались без підключення вебдодатків до мережі. Варто додати, щоокрім тестування усіх модулів клієнтського OT ядра, був розроблений набір тестів для тестування запитів передачі комітів (зліпків даних) до репозиторію на сервер операціональних трансформацій. Результат роботи операціональних перетворень можна спостерігати в терміналі користувача при взаємодії OT ядра з OT сервером, або ж в програмі graphviz.io для онлайн відображення серверних графів.

ВИСНОВКИ

Головною метою даного дипломного проєктує створенняпрограмного забезпечення для клієнтської сторони вебдодатків, яке дозволяло б легко синхронізувати та координувати усю роботу з даними від кількох користувачів. Система розроблялась з допомогою сучасних технологій веброзробки, були проаналізовані та розглянуті їх переваги над класичними методами і підходами. Для надійної роботи з даними в офлайні, а також уникнення серверних конфліктів зберігання даних були використані усі особливості операціональних трансформацій. Таким чином вдалось подолати розповсюджені проблеми в існуючих програмних рішеннях.

Також в ході розробки реалізовано певну базу для побудови вебдодатків, в яких користувачі зможуть зберігати дані будь-якого типу на віддаленому сервері. Дане програмне забезпечення використовує передові технології шифрування даних та є надзвичайно надійним та доступним. Однією з головних переваг такого ПЗ є його швидкодія.

Одним з варіантів подальшого розвитку проєкту є розширення можливих функціональних можливостей вебдодатків, побудованих на основі системи операціональних трансформацій.

Загалом, розробленеклієнтське програмне забезпечення продемонструвало свою надійність та здатність до існування, що можна спостерігати під час його тестування та передачі даних будь-якого типу між користувачем та серверомопераціональних трансформацій.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. БауманА., Бахман Р., Даганд Р., ХаррісТ., Сінганья А. – "In Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles" - Big Sky, Montana, USA , 2009.– ст.11–14.
2. Аншина М. - Технологии создания распределенных систем. Для профессионалов \ М. Аншина. – СПб. , Питер, 2013. - 576 с.
3. Таненбаум Е. - Распределенные системы. Принципы и парадигмы \ Е. Таненбаум, М. Ван Стен. – «Питер», 2014. - 880 с.
4. ГалагерГ., ХамблетП., Спіра П.-"A Distributed Algorithm for Minimum-Weight Spanning Trees" - ACM Transactions on Programming Languages and Systems, 1983. – ст. 66–77.
5. Сан Д., Ксайа С.,Чен Д. - "Operational transformation for collaborative word processing". - Proc. of the ACM Conf. on Computer-Supported Cooperative Work, 2004. - ст. 437-446.
6. Сан С., Еліс С. - "Operational transformation in real-time group editors: issues, algorithms, and achievements". Proceedings of the 1998 ACM conference on Computer supported cooperative workby ACM Press New York, NY, USA, 1998. – ст. 59-68.
7. Геральд Остер,Паскаль Молі ,Паскаль Урсо–"Tombstone Transformation Functions for Ensuring Consistency in Collaborative Editing Systems" – Procs. 2nd Intl. Conf. on Collaborative Computing: Networking, Appln. and Worksharing, 2006.– ст. 85-93.
8. Гу-НінгЯнг, ДжамінгДжанг- "Consistency maintenance based on the mark and retrace technique in groupware systems" - Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work, 2005. – ст. 264-273.
9. КрістоферР.,ПалмерГ., ГордонВ.,КормакВ. - "Operation transforms for a distributed shared spreadsheet. Incorporating advanced collaboration

capabilities into 3D digital media design tools". - Proc. of ACM Conf. on Computer-Supported Cooperative Work, 1998. – ст. 5-8.

10. Руї Лі,Ду Лі - "New Operational Transformation Framework for Real-Time Group Editors" — IEEE Press, 2007. — ст. 307-319.
11. Руї Лі,Ду Лі - "An Admissibility-Based Operational Transformation Framework for Collaborative Editing Systems" - Computer Supported Cooperative Work (CSCW), 2010. – ст. 1–43
12. Мелор-КрумейД.,Скот М. -"Algorithms for scalable synchronization on shared-memory multiprocessors." - ACM Trans. Comput. Syst. 9, 1 1991. – ст. 21–65
13. Мішель Джонсон, Кністер. - "A framework for undoing actions in collaborative systems". - ACM Trans. Comput.-Hum. Interact. 1994. – ст.295–330.
14. Дюн Сан - "Context-based Operational Transformation for Distributed Collaborative Editing Systems". - IEEE TransactionsonParallelandDistributedSystems. 2010 -ст.1454–1470
15. Абдесамад Молі. - "Real time group editors without Operational transformation" – ACM Trans. Comput.-Hum, 2005 – ст. 40-42.
16. Неіл Фразер. - "Differential Synchronization", 2009 – ст. 78-88.
17. Руї Лі. - "Preserving Operation Effects Relation in Group Editors". Proceedings of the ACM CSCW'04 Conference on Computer-Supported Cooperative Work. ACM PressNewYork, NY, USA., 2004 - ст. 457–466.
18. ХаусманДж. "Cost-efficient parallel processing of irregularly structured problems in cloud computing environments". JournalofClusterComputing., 2009 – ст. 887–909
19. Шнейдер В., Ватерхофен Л. - "Trading Bit, Message, and Time Complexity of Distributed Algorithms". In Peleg, D. (ed.). Distributed Computing. Springer Science & Business Media, 2016 -ст. 51–65

20. ТумаріанН.,БархенДж. -"Neural Networks for Real-Time Robotic Applications". In Fijany, A.; Bejczy, A. (eds.). Parallel Computation Systems For Robotics: Algorithms And Architectures. WorldScientific, 1992 - ст. 214.